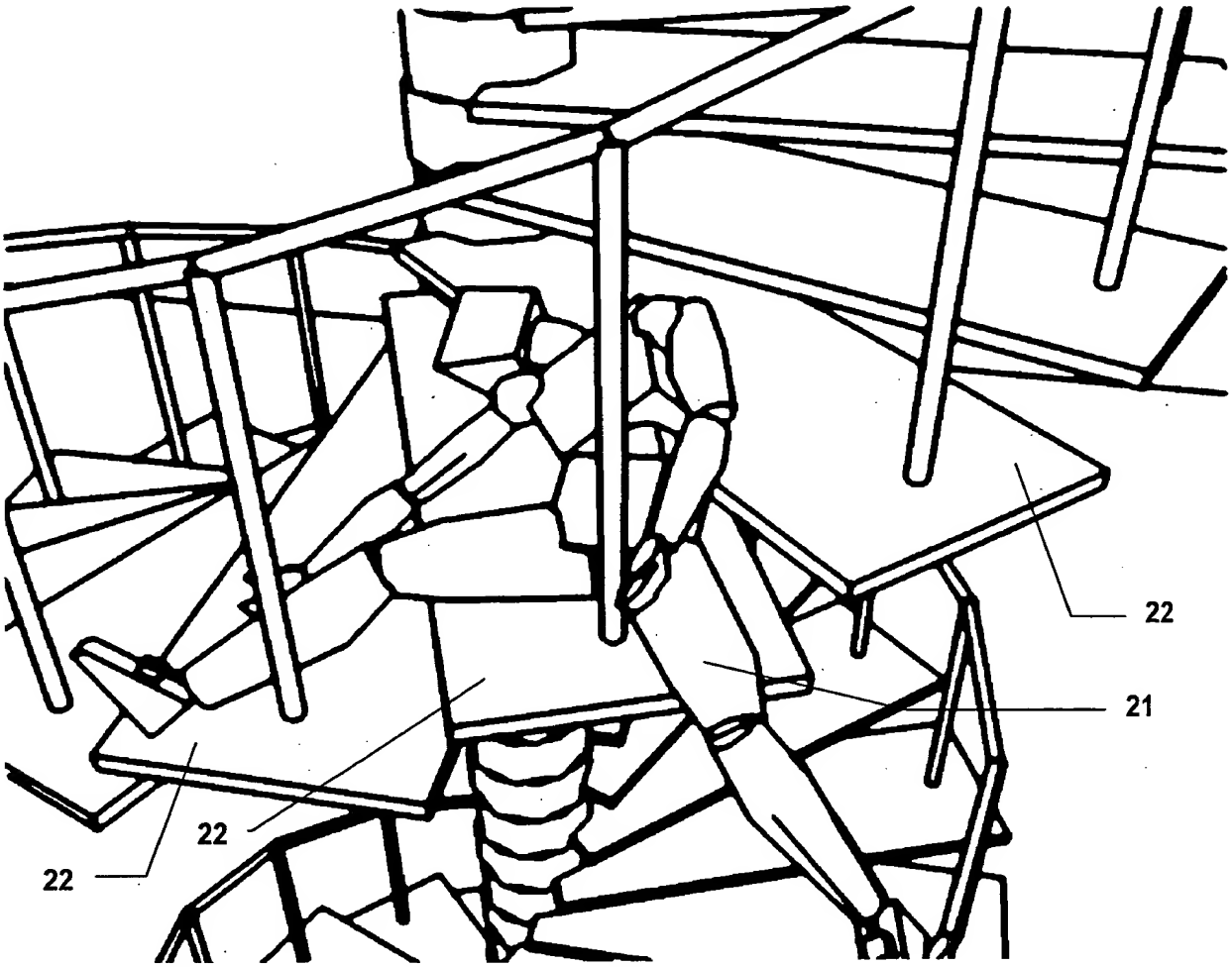


*Fig. 1*

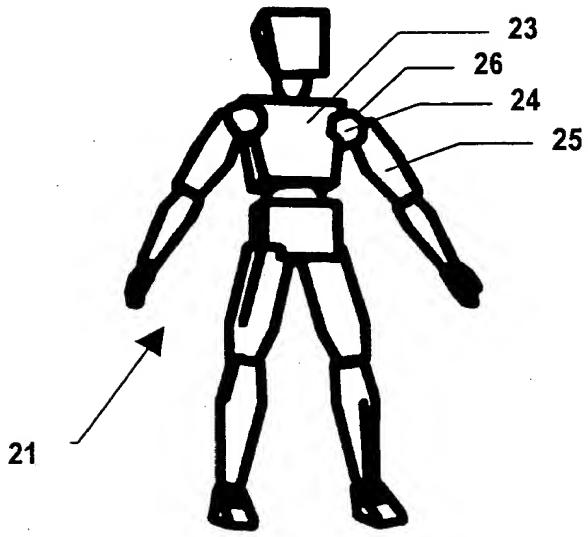


*Fig. 2A*

| JOINT NAME | PARENT JOINT | OFFSET FROM PARENT | MASS | INERTIA MATRIX | JOINT HIGH LIMIT | JOINT LOW LIMIT |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

31 32 33 34 35 36 37

*Fig. 2B*

| BODY NAME | OWNING JOINT | OUTLINE POINTS |
|---|---|---|
| | | |
| | | |
| | | |

38 39 40

*Fig. 2C*

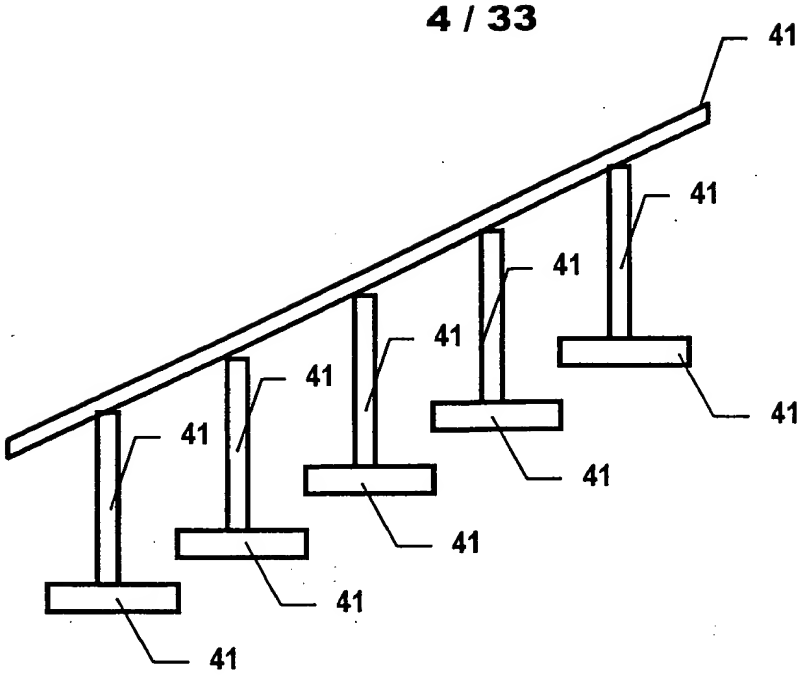


*Fig. 3A*

| BODY NAME | OUTLINE POINTS |
|---|---|
| | |
| | |
| | |

42 43

*Fig. 3B*

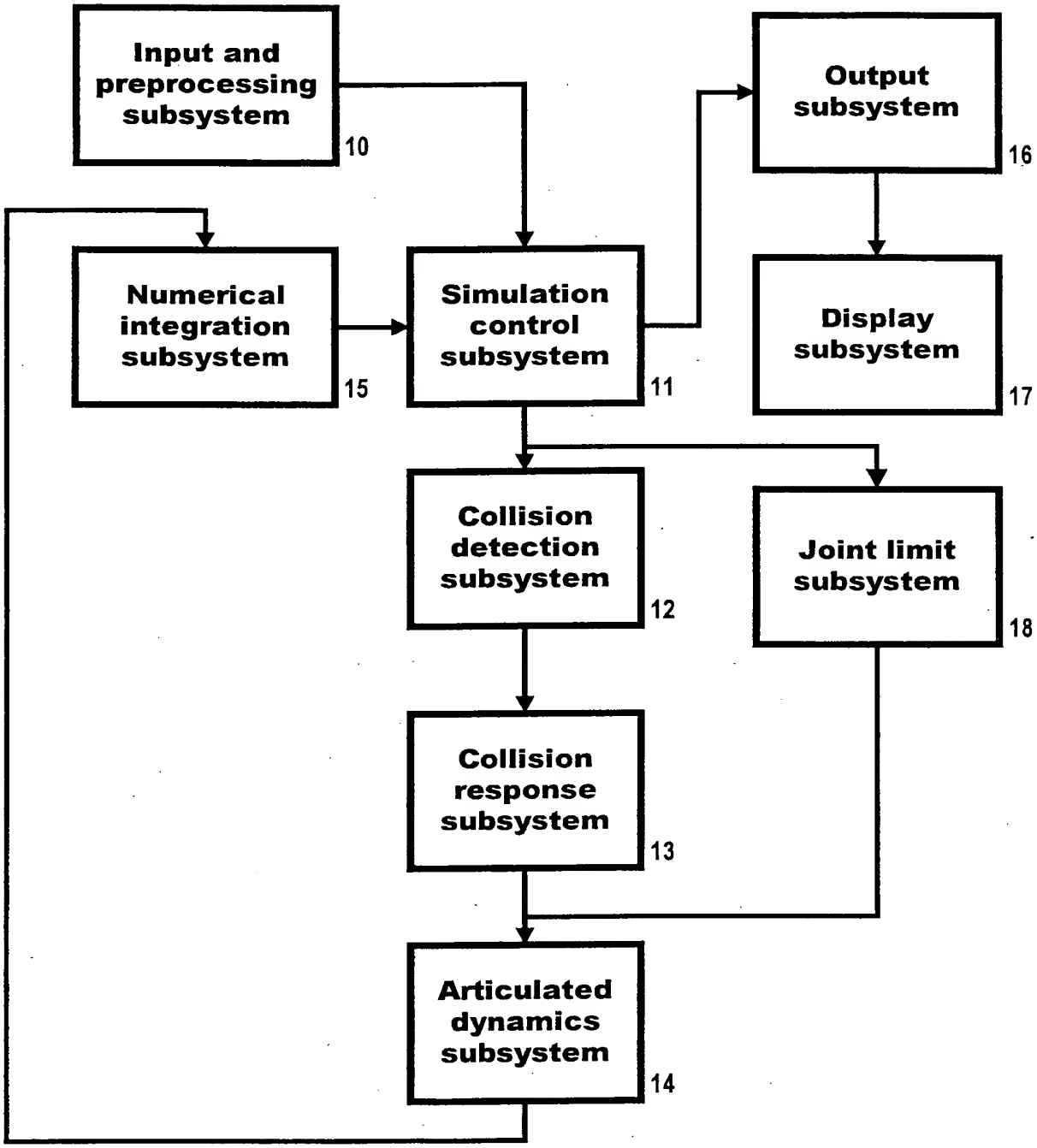
Input and preprocessing subsystem
10
Output subsystem
16
Numerical integration subsystem
15
Simulation control subsystem
11
Display subsystem
17
Collision detection subsystem
12
Joint limit subsystem
18
Collision response subsystem
13
Articulated dynamics subsystem
14

*Fig. 4*

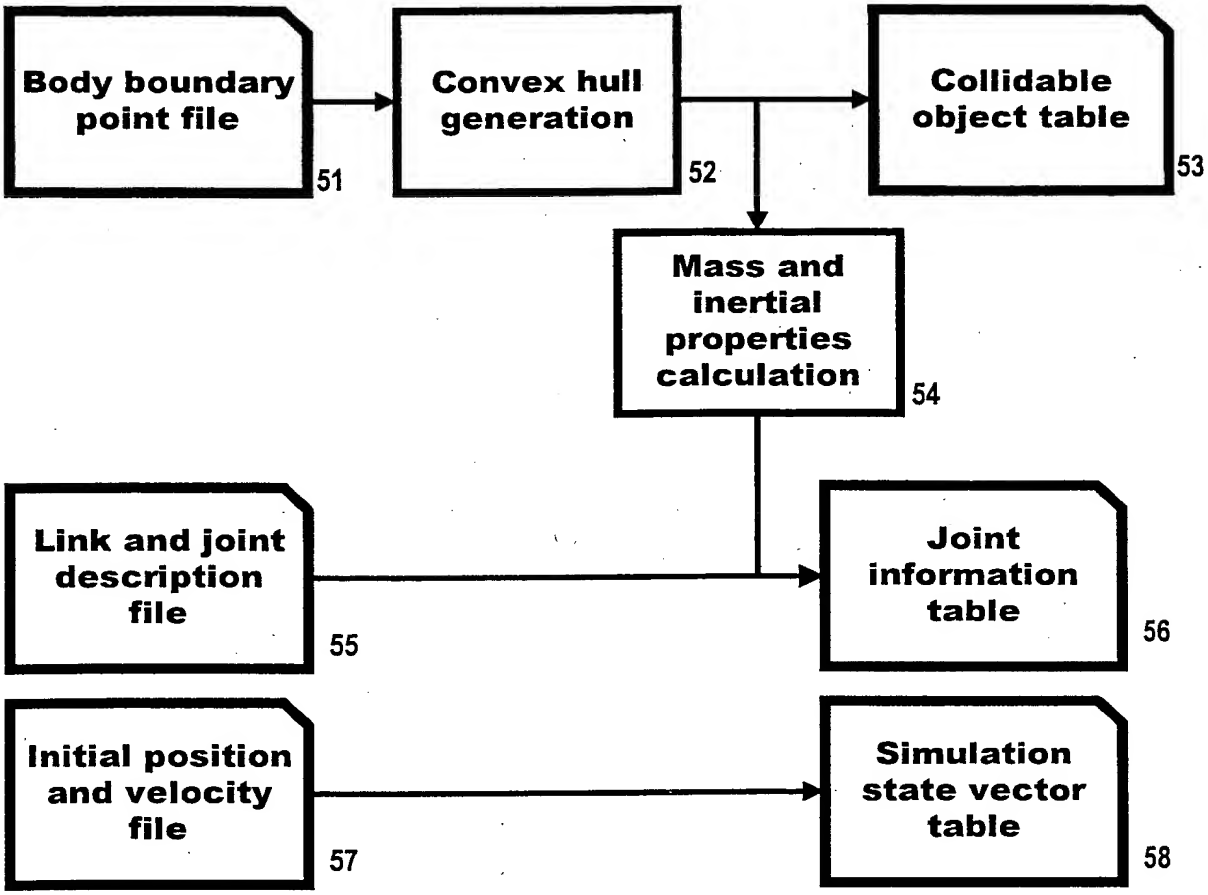
Body boundary point file
51
Convex hull generation
52
Collidable object table
53
Mass and inertial properties calculation
54
Link and joint description file
55
Joint information table
56
Initial position and velocity file
57
Simulation state vector table
58

*Fig. 5*

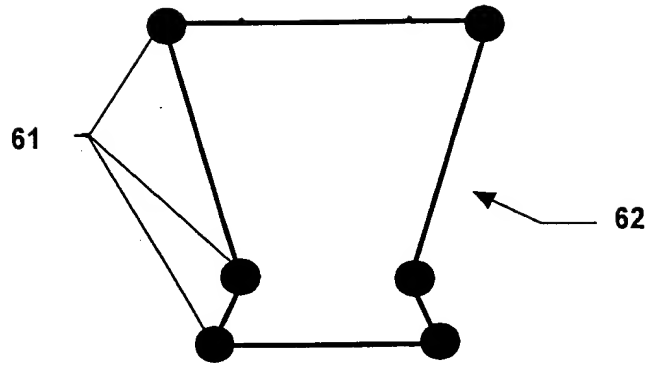


*Fig. 6A*

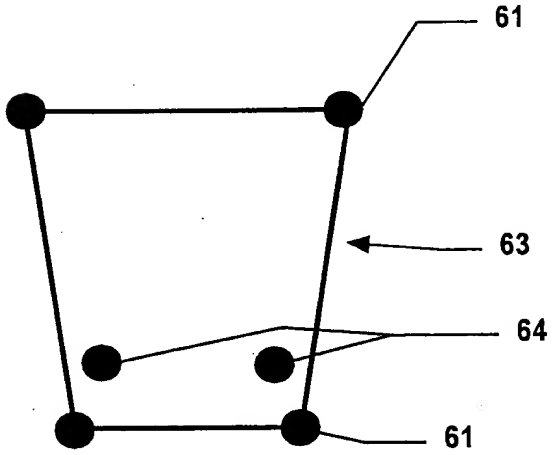


*Fig. 6B*

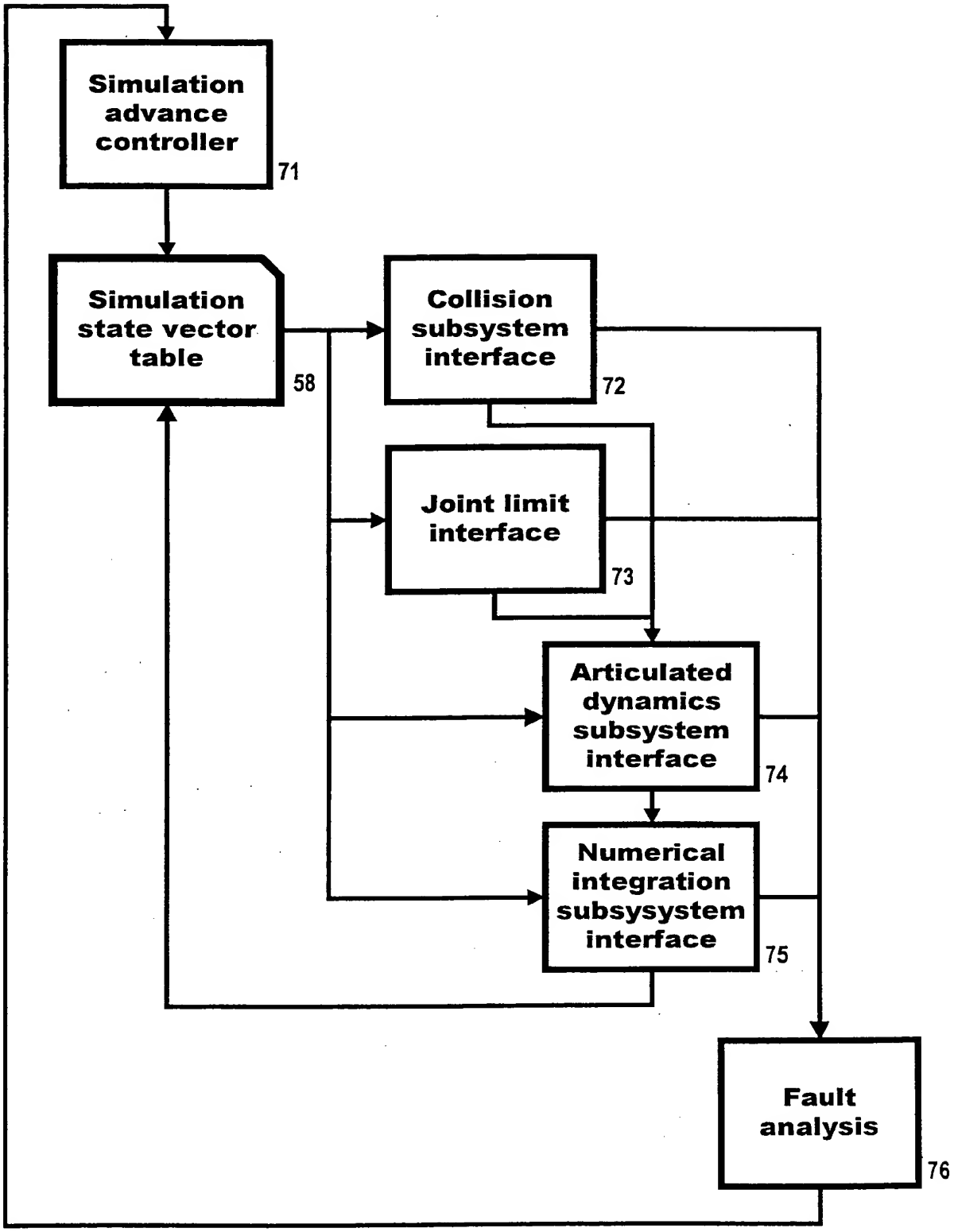
Simulation advance controller
71
Simulation state vector table
58
Collision subsystem interface
72
Joint limit interface
73
Articulated dynamics subsystem interface
74
Numerical integration subsysystem interface
75
Fault analysis
76

*Fig. 7*

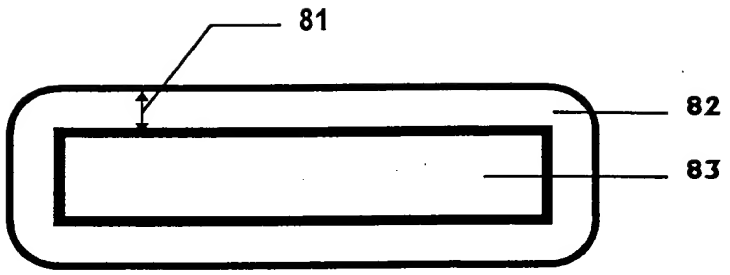


*Fig. 8A*

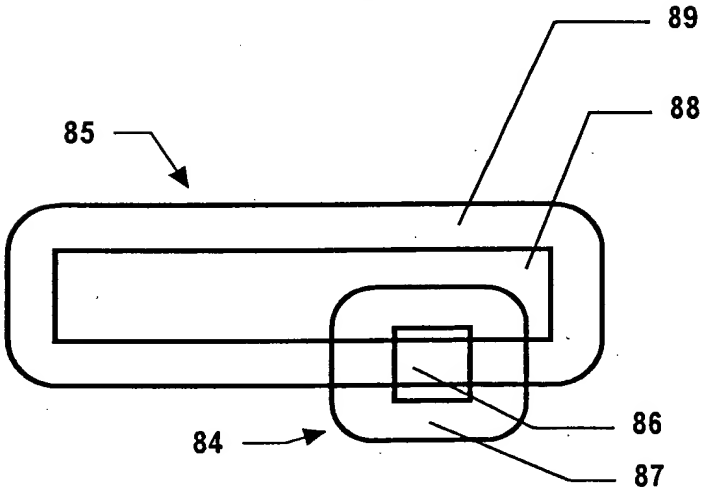


*Fig. 8B*

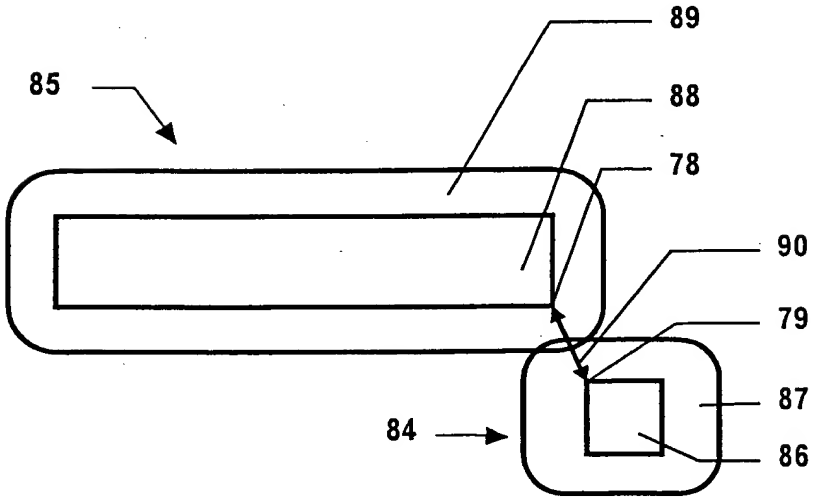


*Fig. 8C*

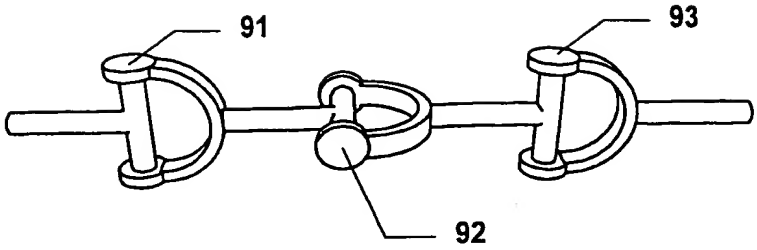


*Fig. 9*

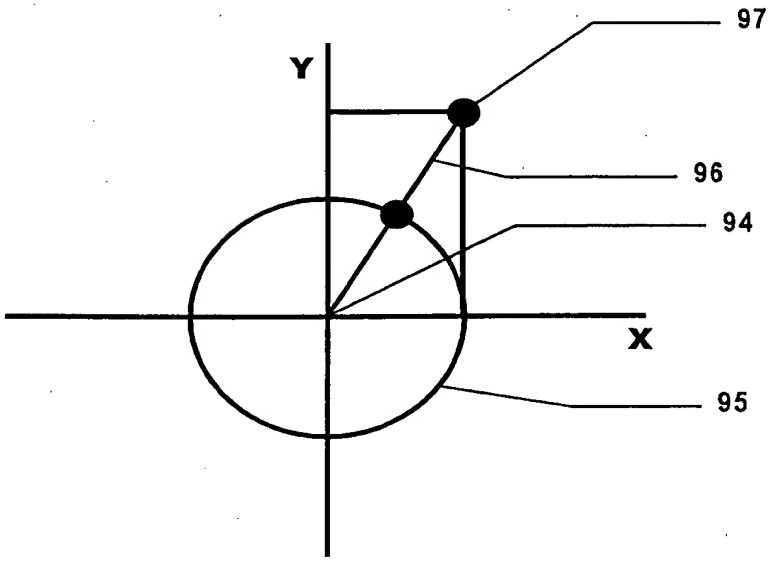


*Fig. 10*

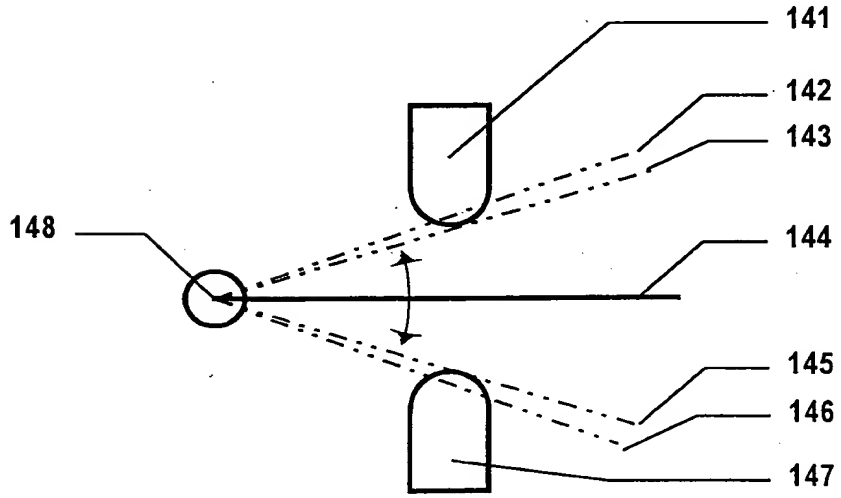


*Fig. 11*

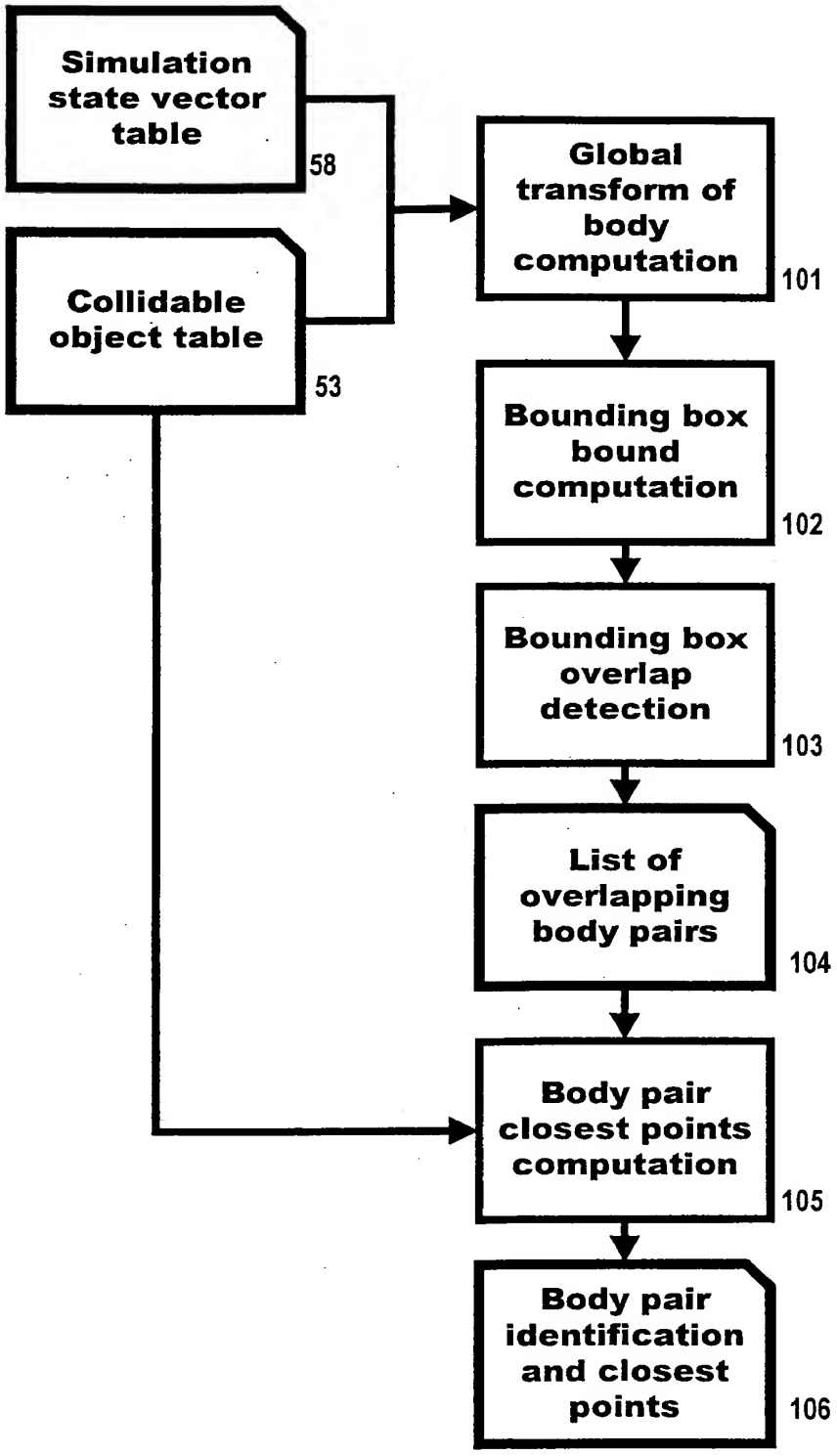
Simulation state vector table
58
Collidable object table
53
Global transform of body computation
101
Bounding box bound computation
102
Bounding box overlap detection
103
List of overlapping body pairs
104
Body pair closest points computation
105
Body pair identification and closest points
106

*Fig. 12*

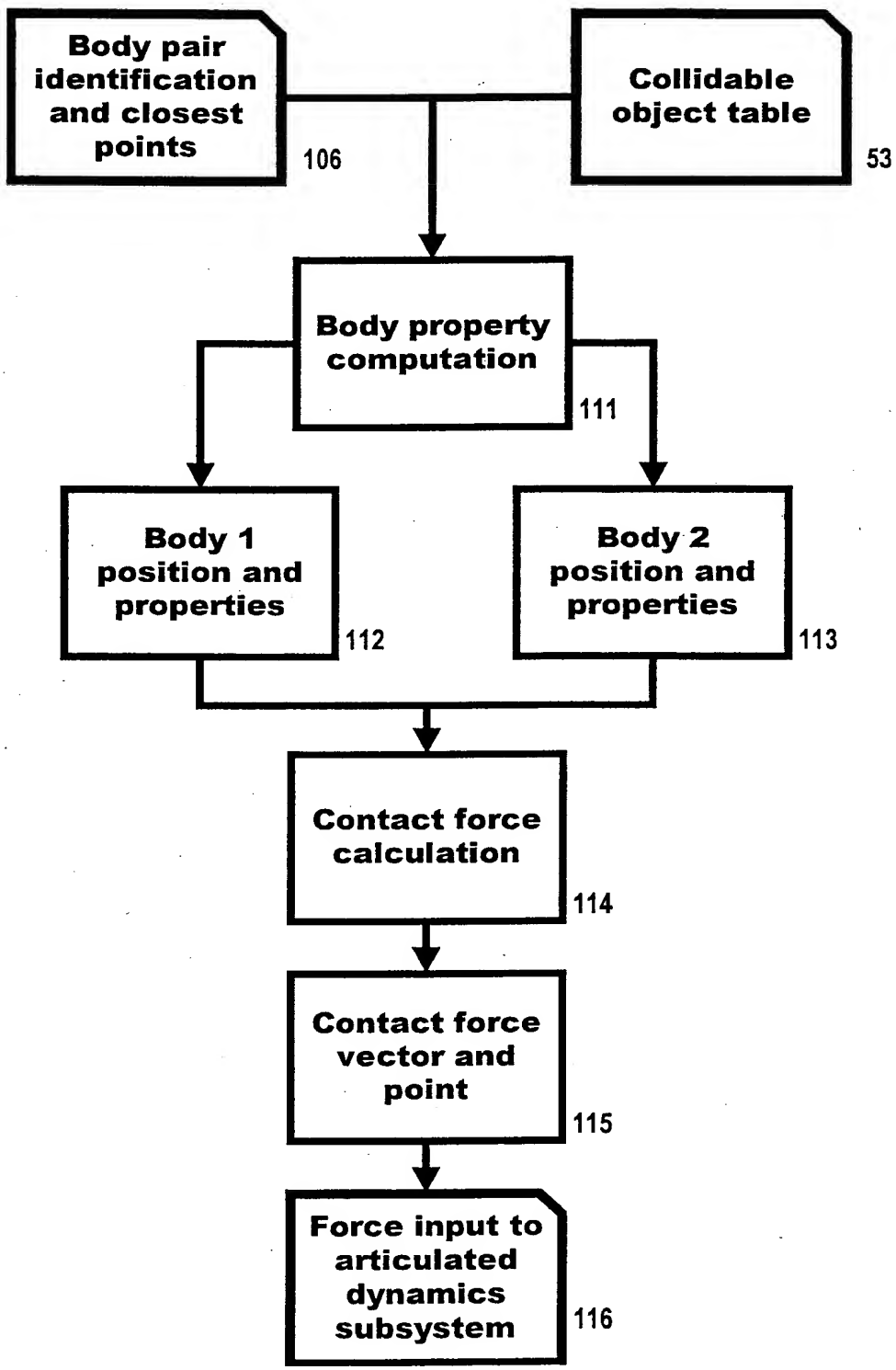
Body pair identification and closest points
106
Collidable object table
53
Body property computation
111
Body 1 position and properties
112
Body 2 position and properties
113
Contact force calculation
114
Contact force vector and point
115
Force input to articulated dynamics subsystem
116

*Fig. 13*

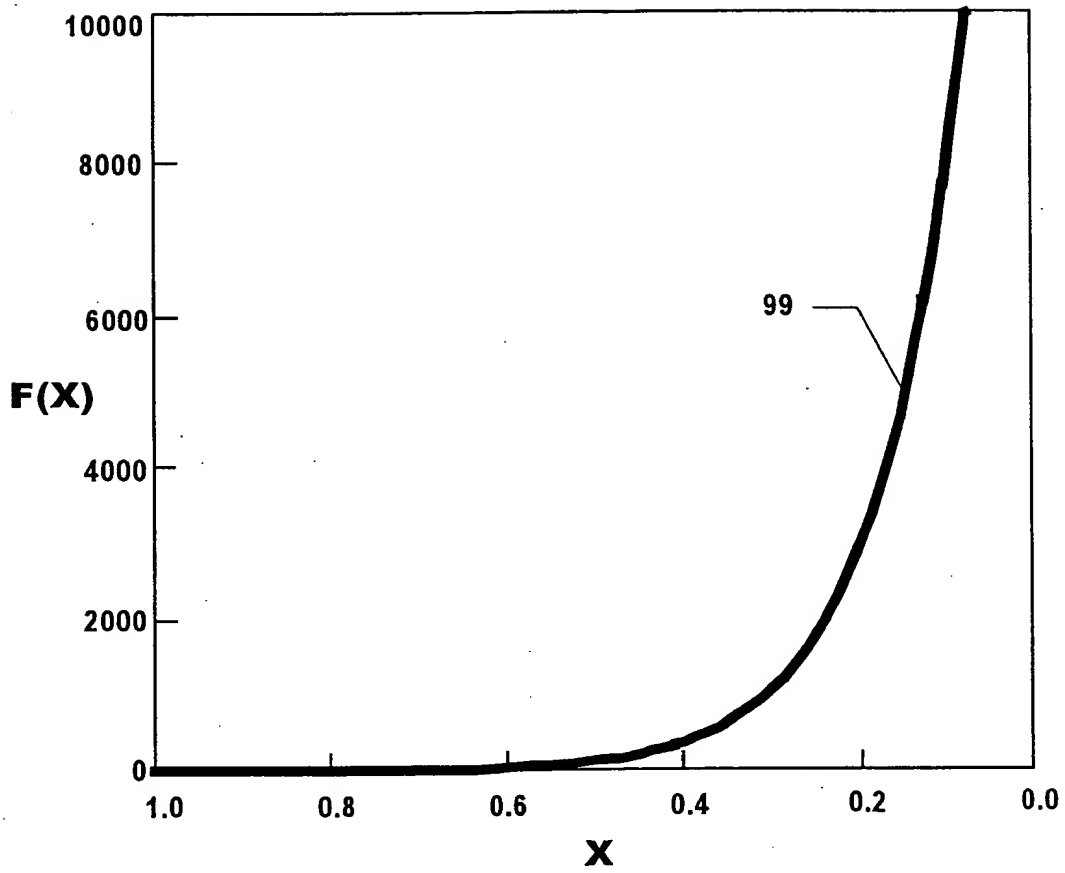
10000
8000
6000
4000
2000
0
F(X)
99
1.0
0.8
0.6
0.4
0.2
0.0
X

*Fig. 14*

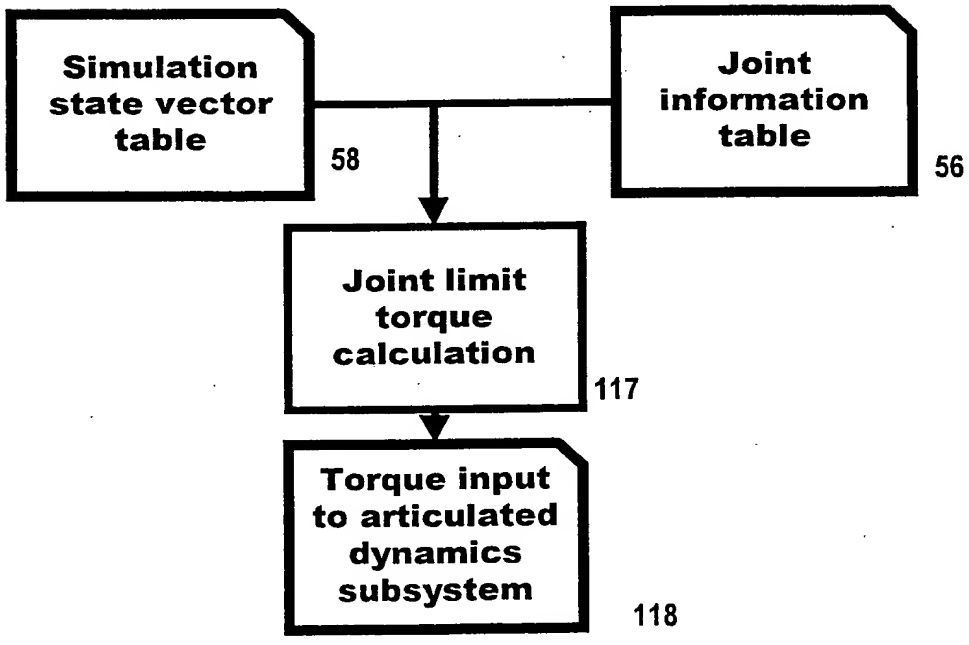
Simulation state vector table
58
Joint information table
56
Joint limit torque calculation
117
Torque input to articulated dynamics subsystem
118

*Fig. 15*

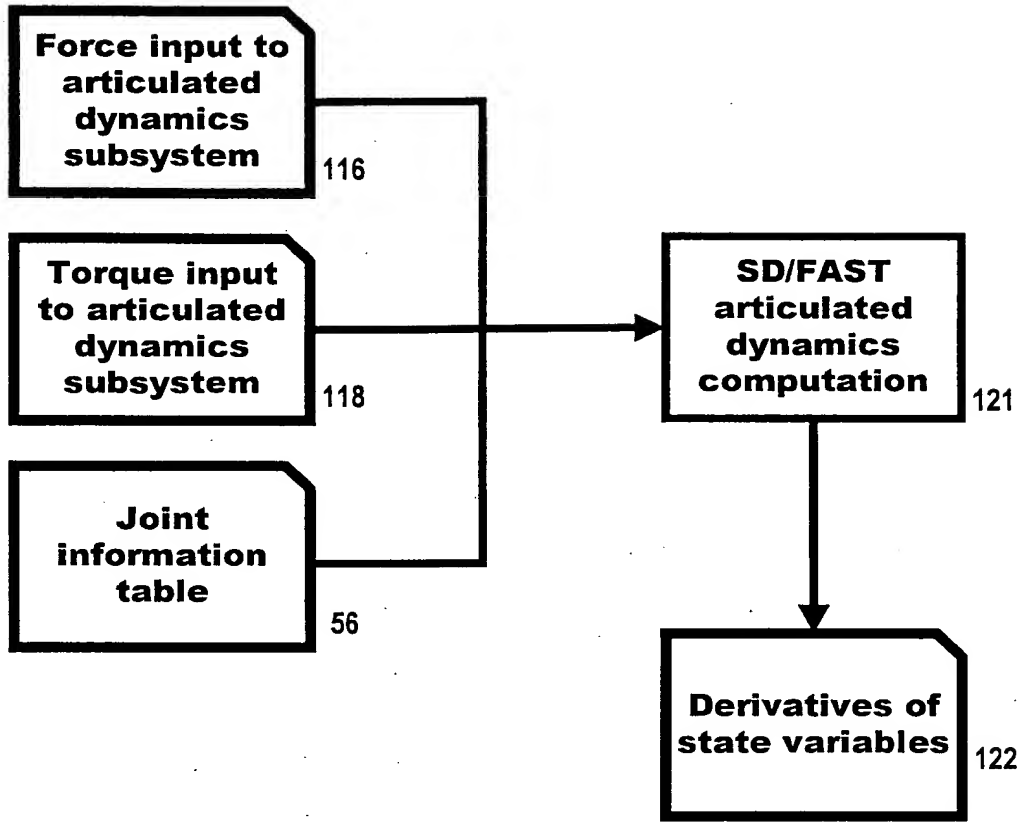
Force input to articulated dynamics subsystem
116
Torque input to articulated dynamics subsystem
118
Joint information table
56
SD/FAST articulated dynamics computation
121
Derivatives of state variables
122

*Fig. 16*

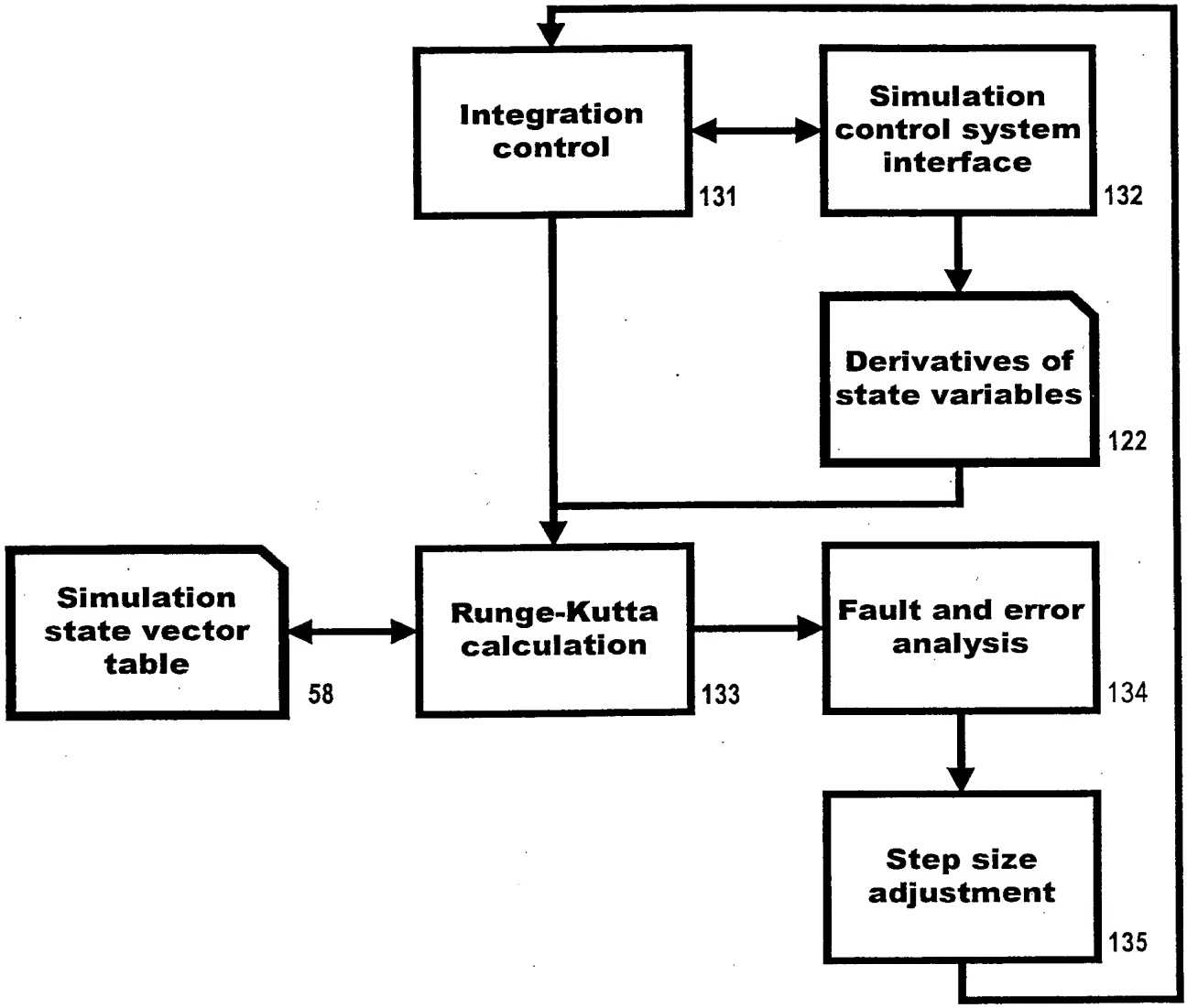
Integration control
131
Simulation control system interface
132
Derivatives of state variables
122
Simulation state vector table
58
Runge-Kutta calculation
133
Fault and error analysis
134
Step size adjustment
135

*Fig. 17*

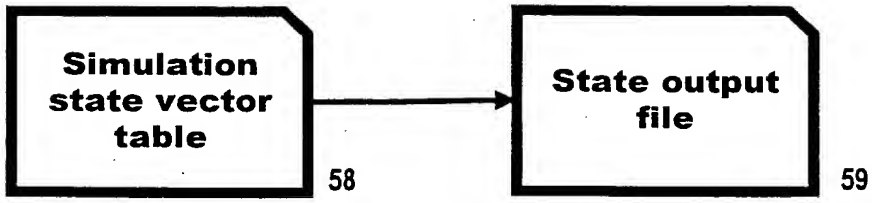
Simulation state vector table
58
State output file
59

*Fig. 18*

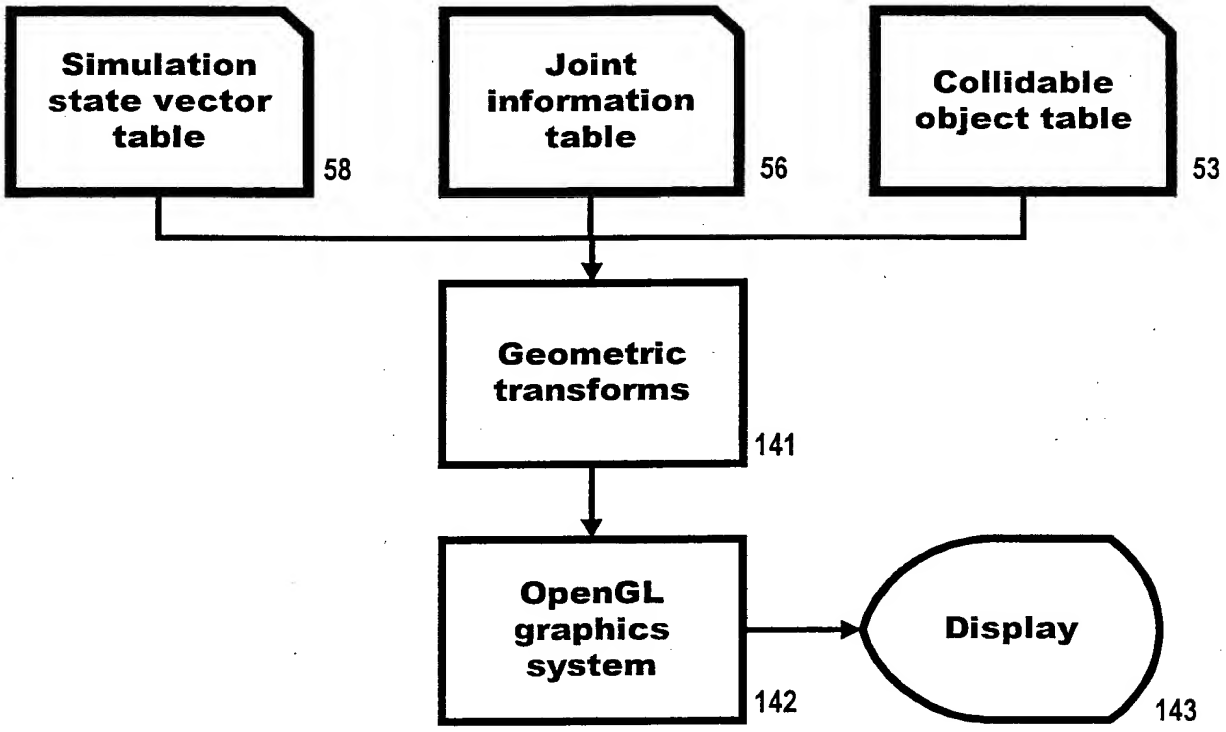
Simulation state vector table
58
Joint information table
56
Collidable object table
53
Geometric transforms
141
OpenGL graphics system
142
Display
143

*Fig. 19*

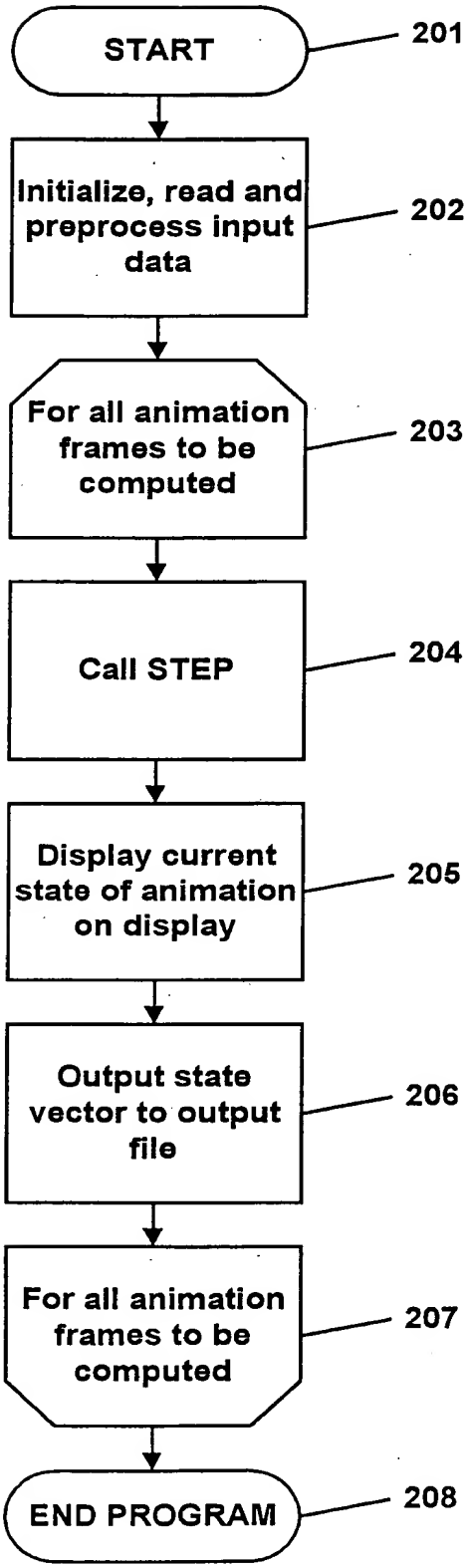
START
201
Initialize, read and preprocess input data
202
For all animation frames to be computed
203
Call STEP
204
Display current state of animation on display
205
Output state vector to output file
206
For all animation frames to be computed
207
END PROGRAM
208

*Fig. 20*

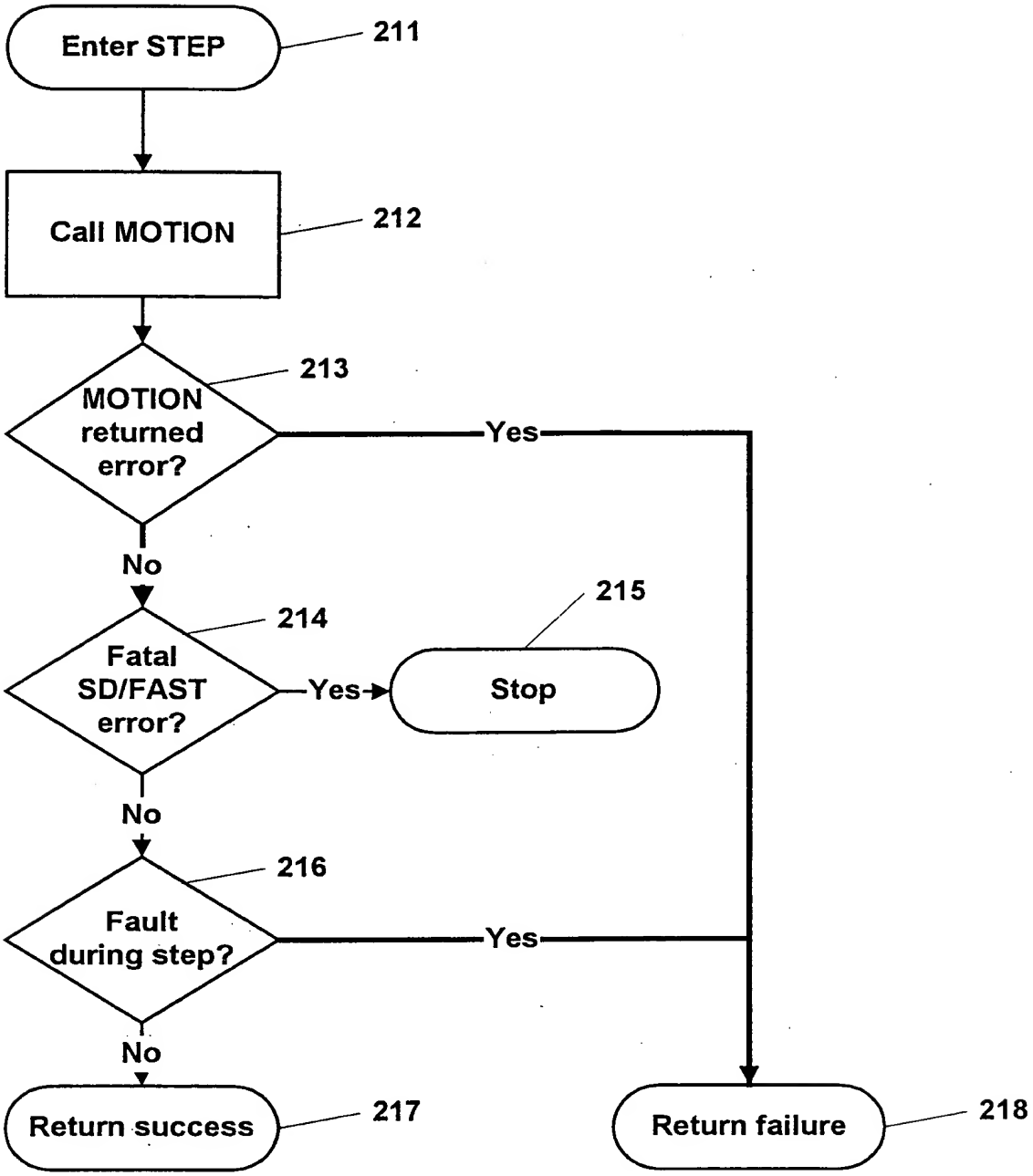
Enter STEP
211
Call MOTION
212
MOTION returned error?
213
Yes
No
Fatal SD/FAST error?
214
Yes
215
Stop
No
Fault during step?
216
Yes
No
Return success
217
Return failure
218

*Fig. 21*

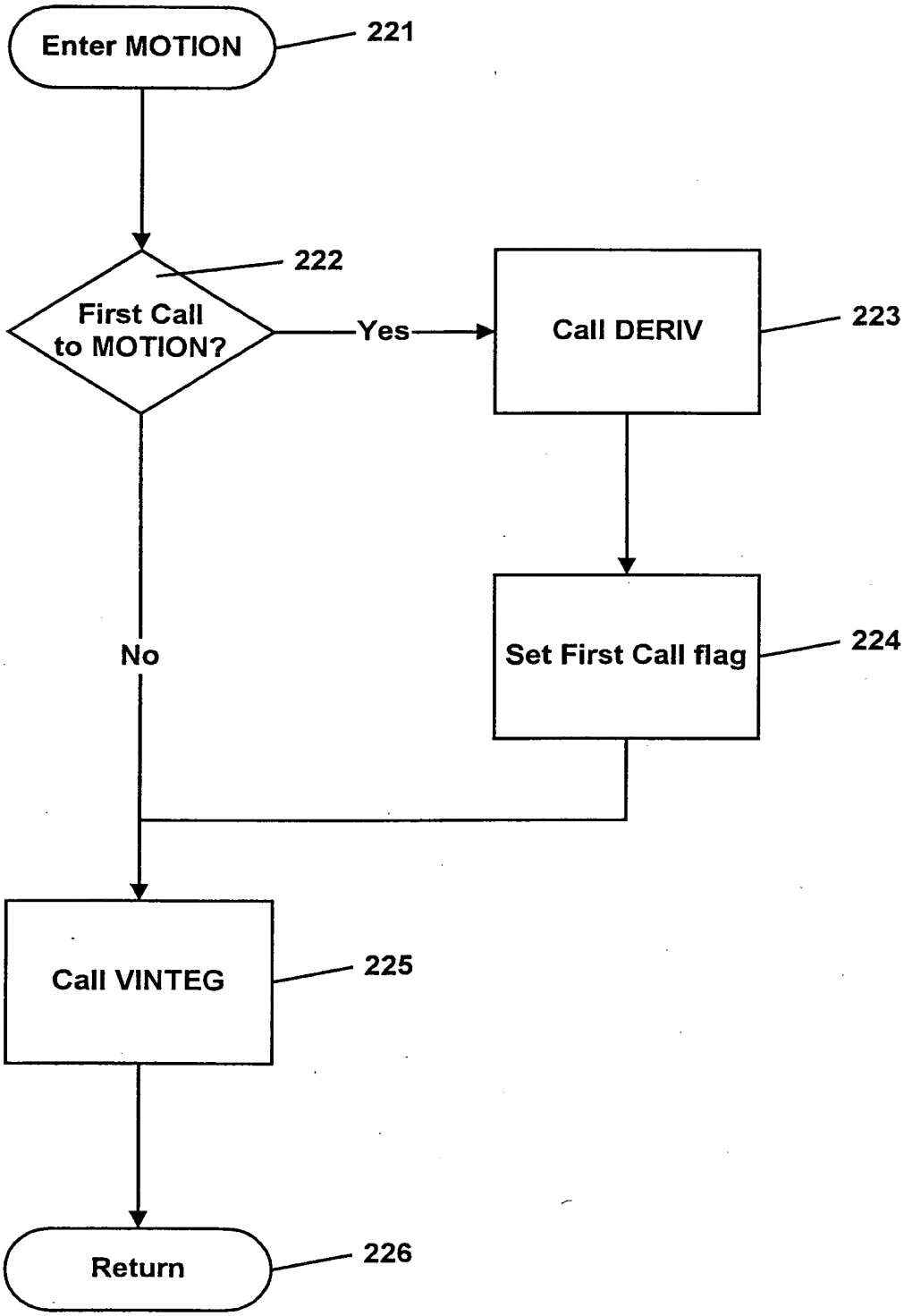
Enter MOTION
221
First Call to MOTION?
222
Yes
Call DERIV
223
Set First Call flag
224
No
Call VINTEG
225
Return
226

*Fig. 22*

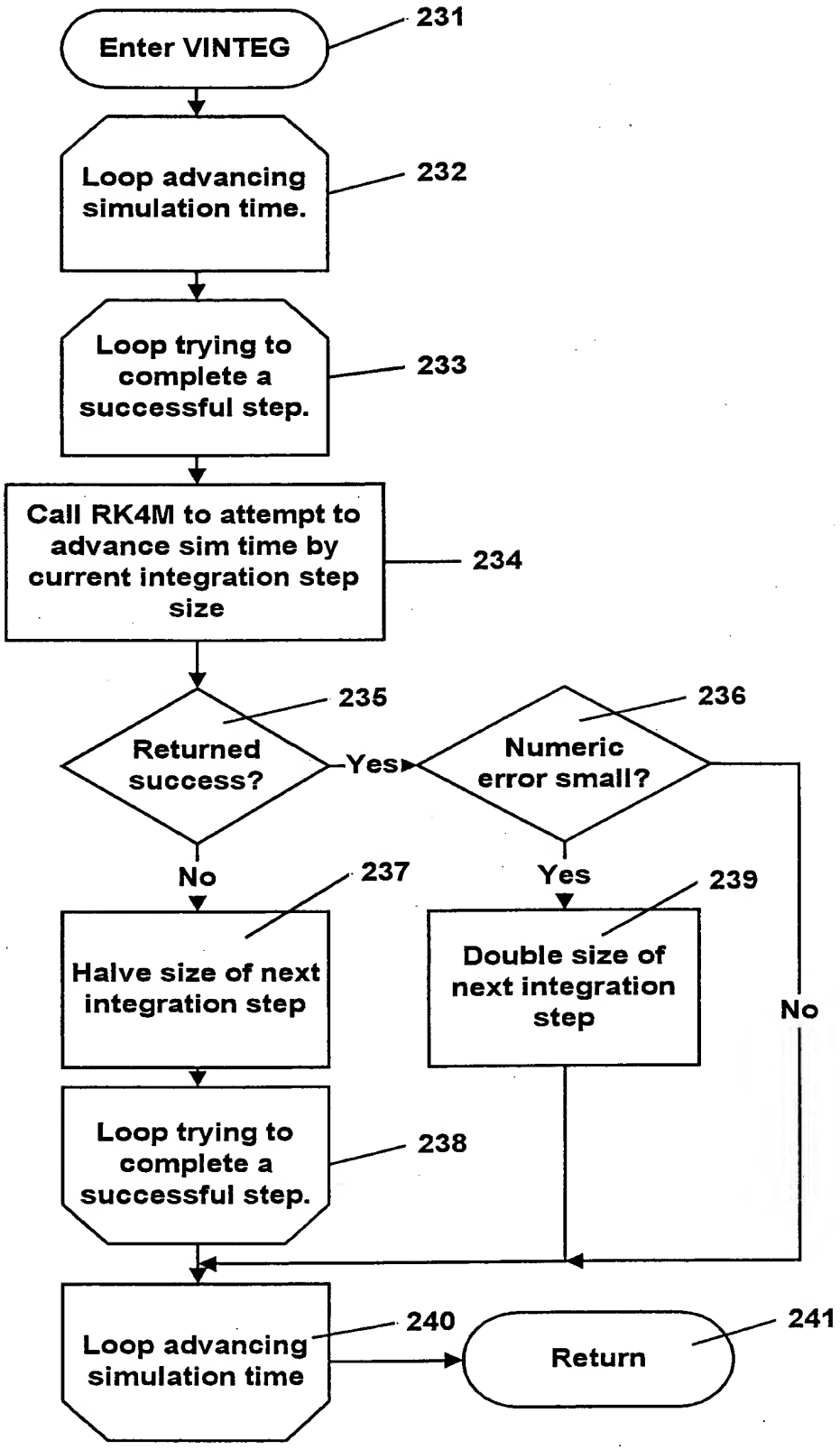
Enter VINTEG
231
Loop advancing simulation time.
232
Loop trying to complete a successful step.
233
Call RK4M to attempt to advance sim time by current integration step size
234
Returned success?
235
Yes
Numeric error small?
236
No
Yes
No
237
239
Halve size of next integration step
Double size of next integration step
Loop trying to complete a successful step.
238
Loop advancing simulation time
240
Return
241

*Fig. 23*

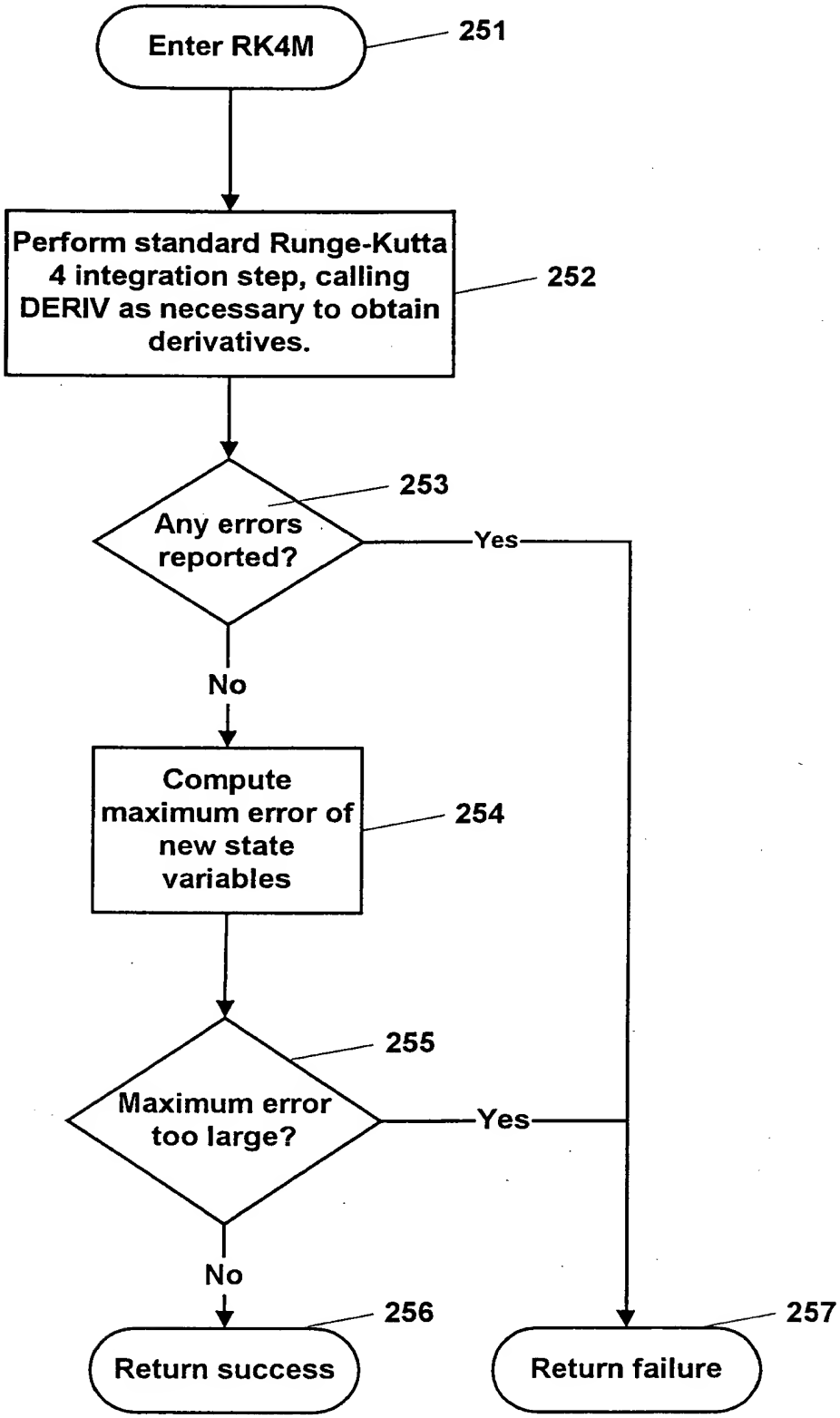
Enter RK4M
251
Perform standard Runge-Kutta 4 integration step, calling DERIV as necessary to obtain derivatives.
252
Any errors reported?
253
Yes
No
Compute maximum error of new state variables
254
Maximum error too large?
255
Yes
No
Return success
256
Return failure
257

*Fig. 24*

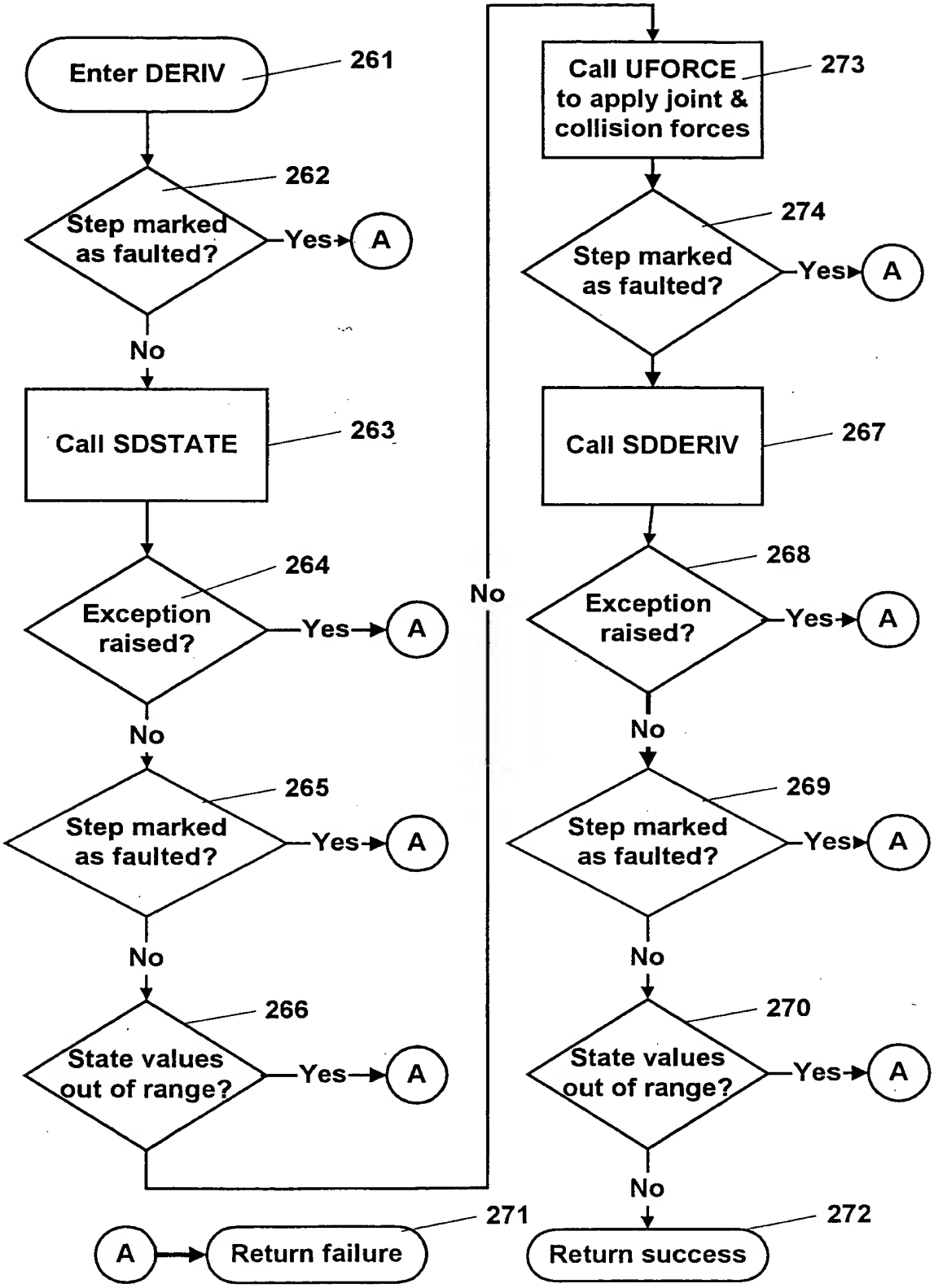
Enter DERIV
261
Step marked as faulted?
262
Yes
A
No
Call SDSTATE
263
Exception raised?
264
Yes
A
No
Step marked as faulted?
265
Yes
A
No
State values out of range?
266
Yes
A
No
Call UFORCE to apply joint & collision forces
273
Step marked as faulted?
274
Yes
A
Call SDDERIV
267
Exception raised?
268
Yes
A
No
Step marked as faulted?
269
Yes
A
No
State values out of range?
270
Yes
A
No
A
Return failure
271
Return success
272

*Fig. 25*

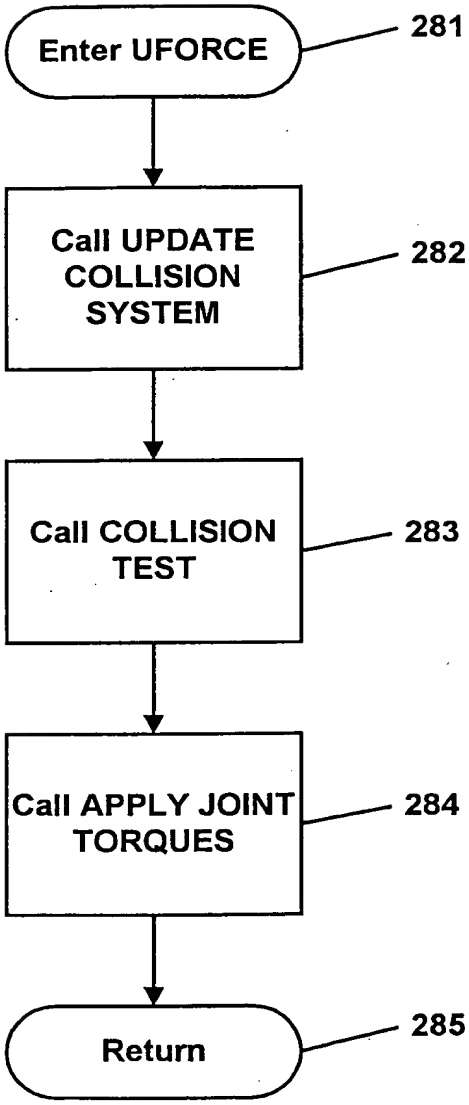
Enter UFORCE
281
Call UPDATE COLLISION SYSTEM
282
Call COLLISION TEST
283
Call APPLY JOINT TORQUES
284
Return
285

*Fig. 26*

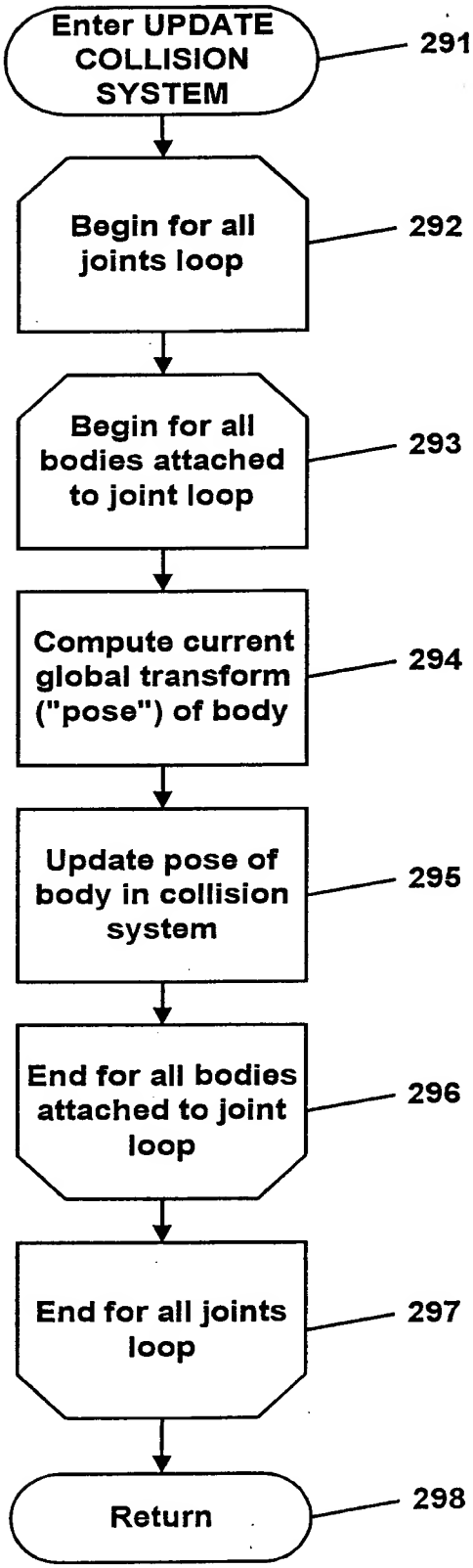
Enter UPDATE COLLISION SYSTEM
291
Begin for all joints loop
292
Begin for all bodies attached to joint loop
293
Compute current global transform ("pose") of body
294
Update pose of body in collision system
295
End for all bodies attached to joint loop
296
End for all joints loop
297
Return
298

*Fig. 27*

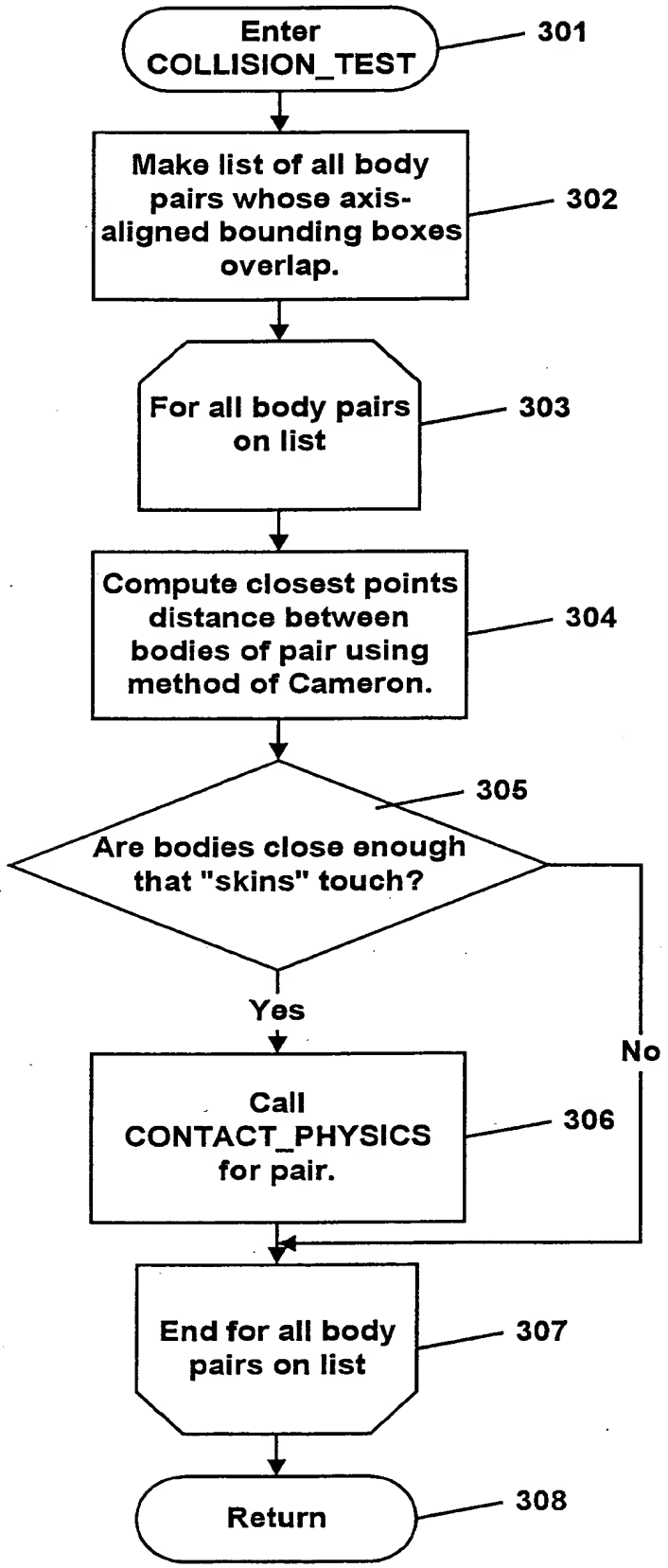
Enter COLLISION_TEST
301
Make list of all body pairs whose axis-aligned bounding boxes overlap.
302
For all body pairs on list
303
Compute closest points distance between bodies of pair using method of Cameron.
304
Are bodies close enough that "skins" touch?
305
Yes
No
Call CONTACT_PHYSICS for pair.
306
End for all body pairs on list
307
Return
308

*Fig. 28*

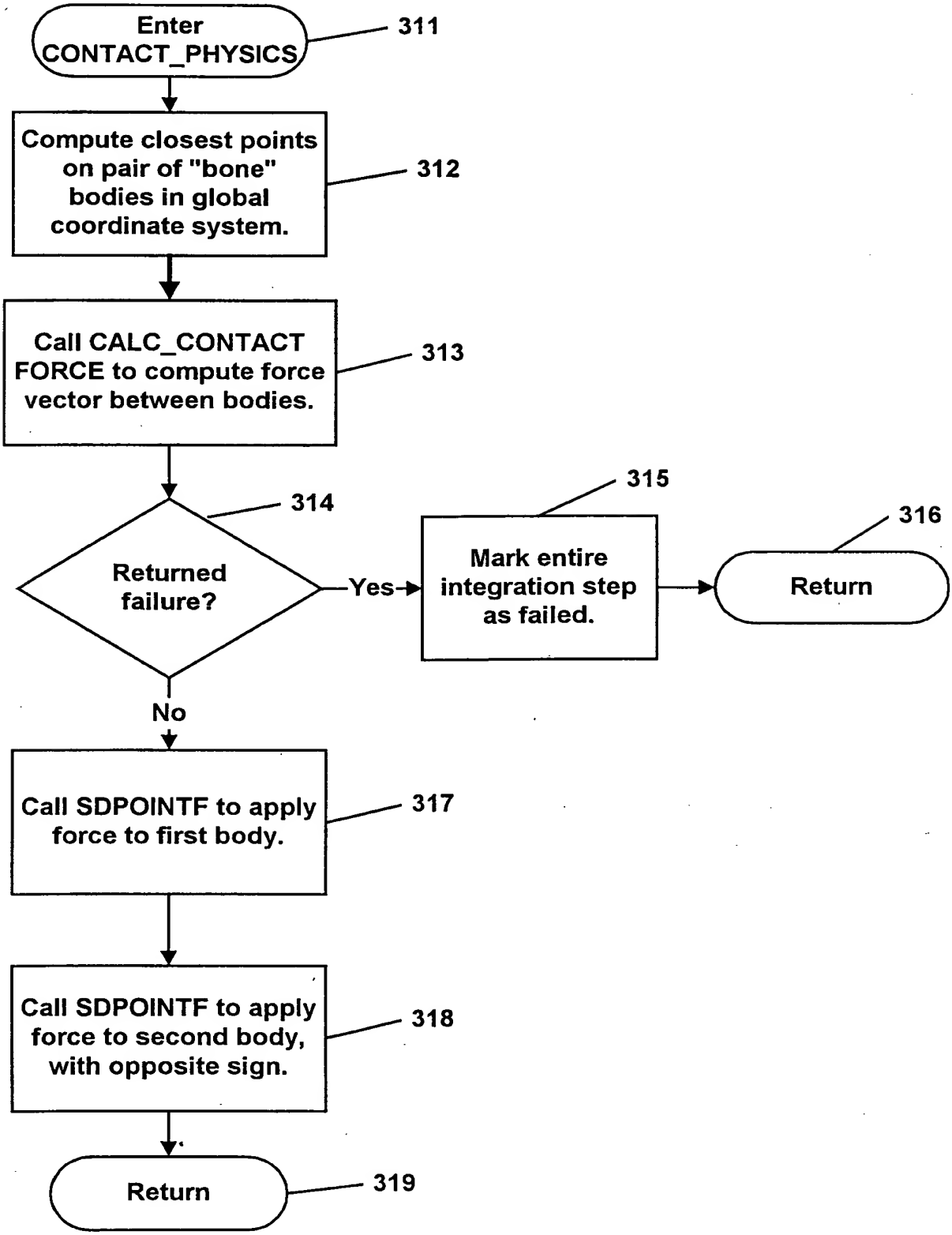
Enter CONTACT_PHYSICS
311
Compute closest points on pair of "bone" bodies in global coordinate system.
312
Call CALC_CONTACT FORCE to compute force vector between bodies.
313
314
Returned failure?
Yes
315
Mark entire integration step as failed.
316
Return
No
Call SDPOINTF to apply force to first body.
317
Call SDPOINTF to apply force to second body, with opposite sign.
318
Return
319

*Fig. 29*

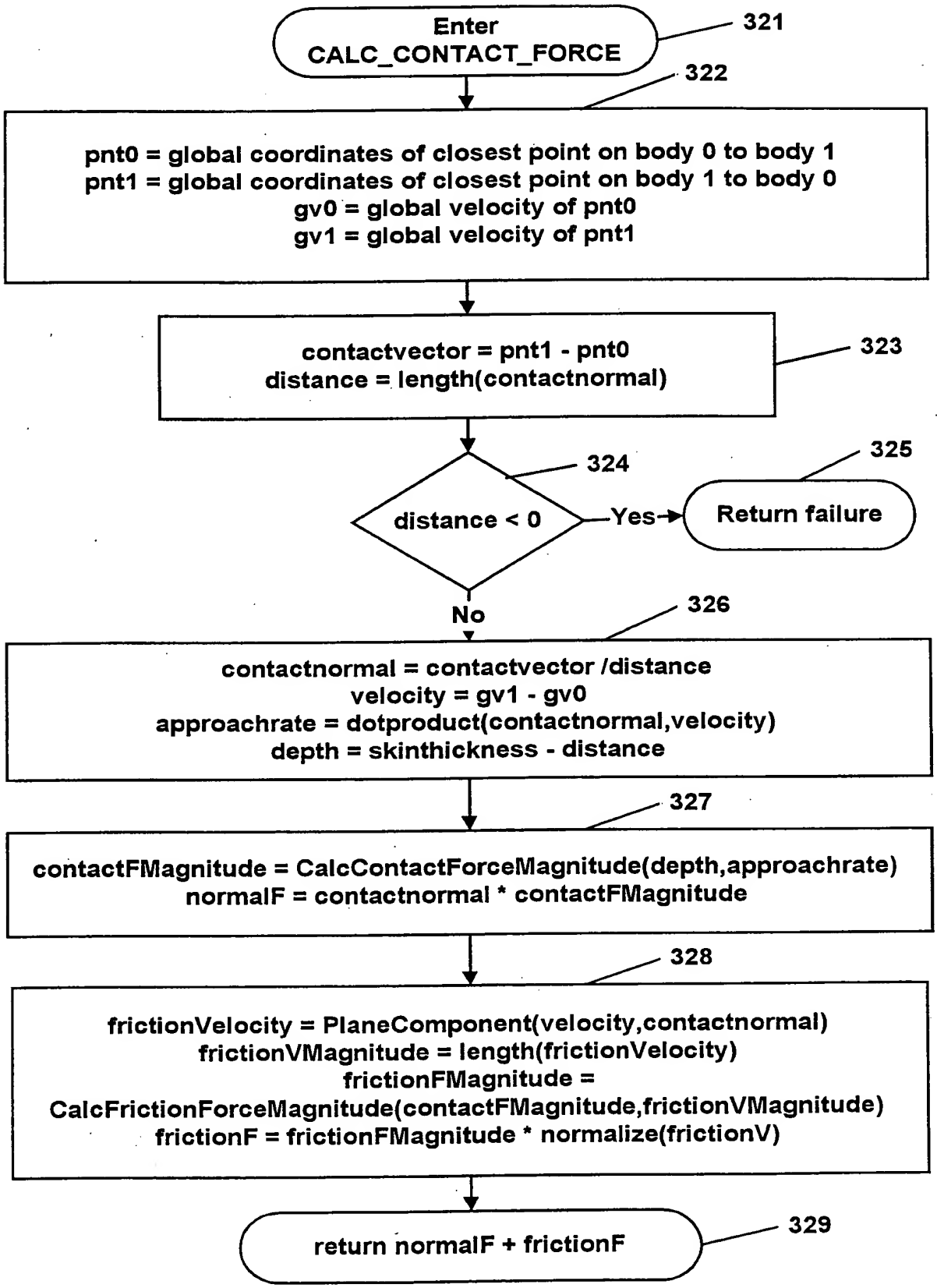
Enter
CALC_CONTACT_FORCE
321
322
pnt0 = global coordinates of closest point on body 0 to body 1
pnt1 = global coordinates of closest point on body 1 to body 0
gv0 = global velocity of pnt0
gv1 = global velocity of pnt1
contactvector = pnt1 - pnt0
distance = length(contactnormal)
323
324
distance < 0
Yes
325
Return failure
No
326
contactnormal = contactvector /distance
velocity = gv1 - gv0
approachrate = dotproduct(contactnormal,velocity)
depth = skinthickness - distance
327
contactFMagnitude = CalcContactForceMagnitude(depth,approachrate)
normalF = contactnormal * contactFMagnitude
328
frictionVelocity = PlaneComponent(velocity,contactnormal)
frictionVMagnitude = length(frictionVelocity)
frictionFMagnitude =
CalcFrictionForceMagnitude(contactFMagnitude,frictionVMagnitude)
frictionF = frictionFMagnitude * normalize(frictionV)
return normalF + frictionF
329

*Fig. 30*

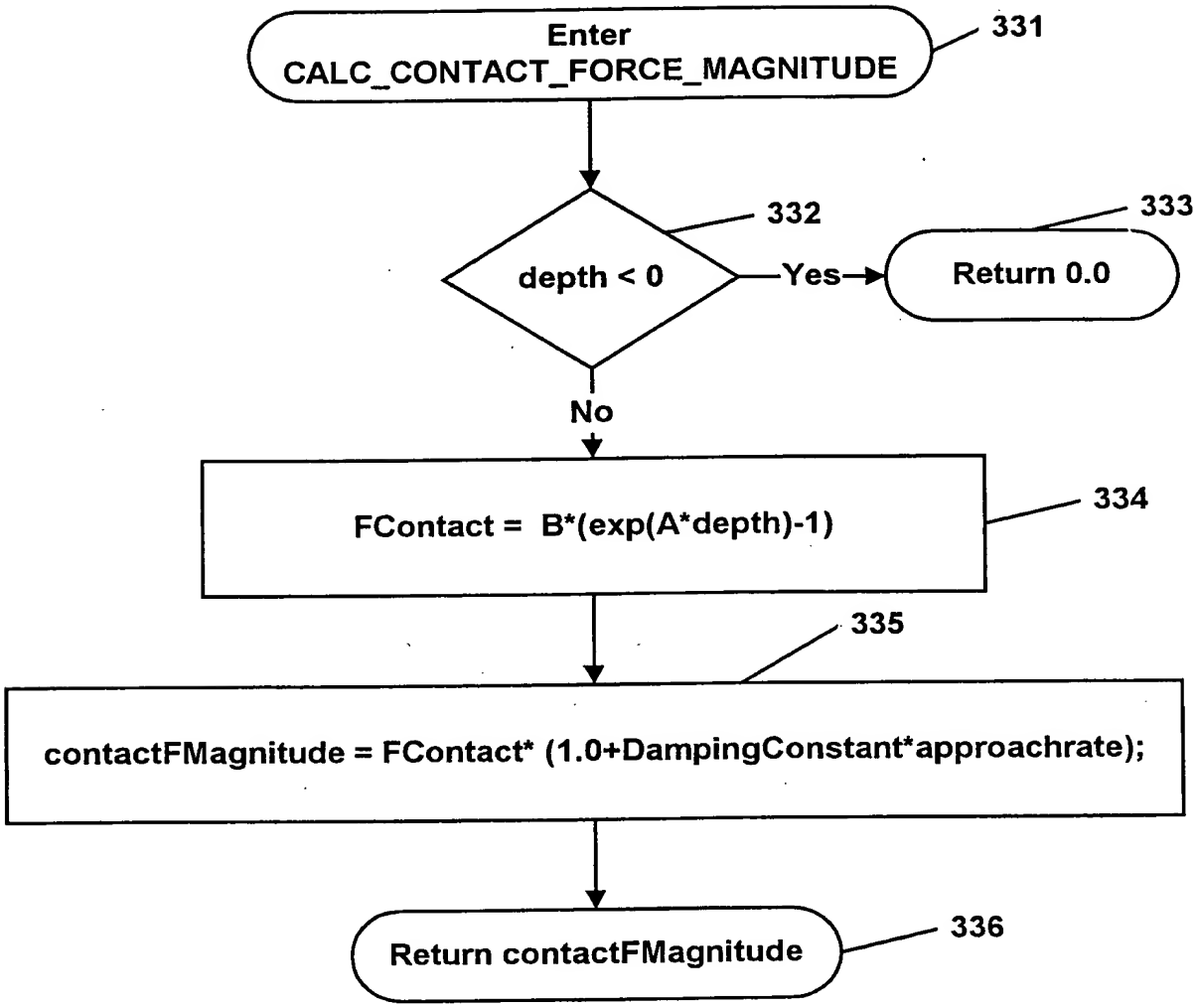
Enter
CALC_CONTACT_FORCE_MAGNITUDE
331
depth < 0
332
Yes
Return 0.0
333
No
FContact = B*(exp(A*depth)-1)
334
335
contactFMagnitude = FContact* (1.0+DampingConstant*approachrate);
Return contactFMagnitude
336

*Fig. 31*

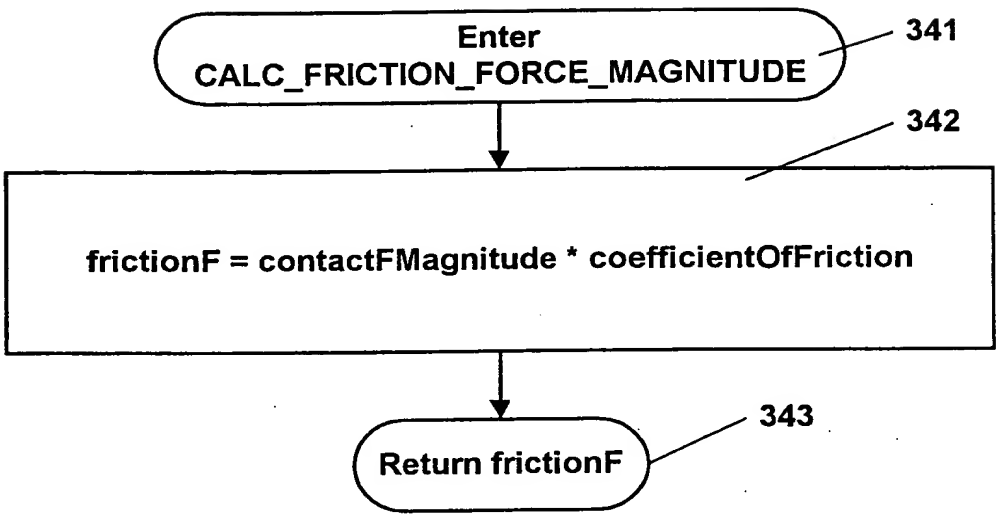
Enter
CALC_FRICTION_FORCE_MAGNITUDE
341
342
frictionF = contactFMagnitude * coefficientOfFriction
343
Return frictionF

*Fig. 32*

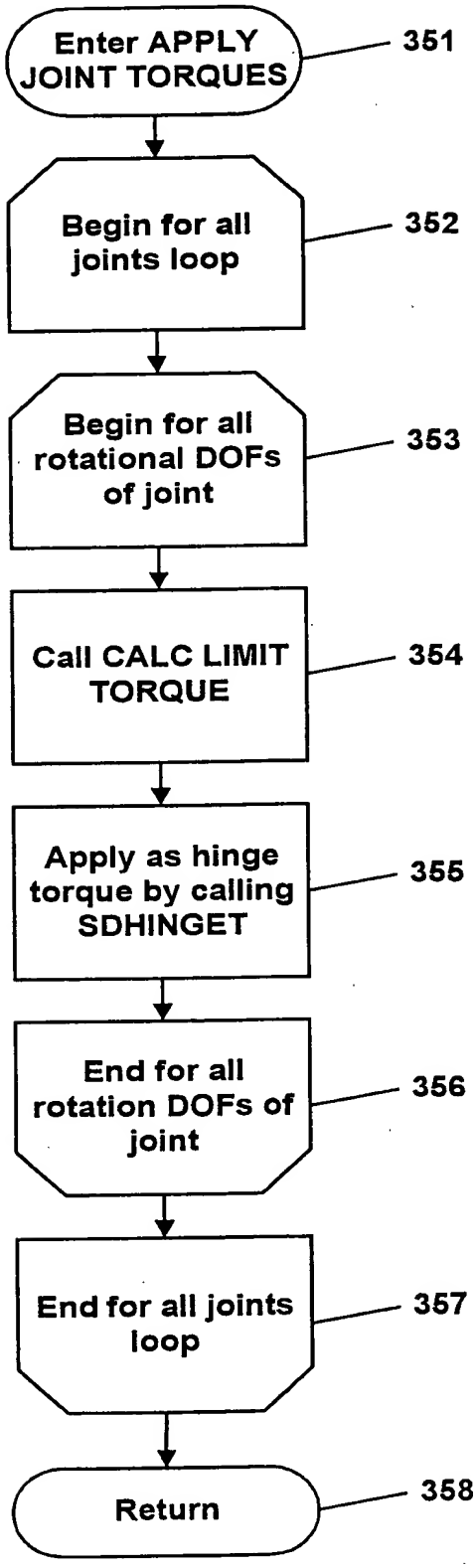
Enter APPLY JOINT TORQUES
351
Begin for all joints loop
352
Begin for all rotational DOFs of joint
353
Call CALC LIMIT TORQUE
354
Apply as hinge torque by calling SDHINGET
355
End for all rotation DOFs of joint
356
End for all joints loop
357
Return
358

*Fig. 33*

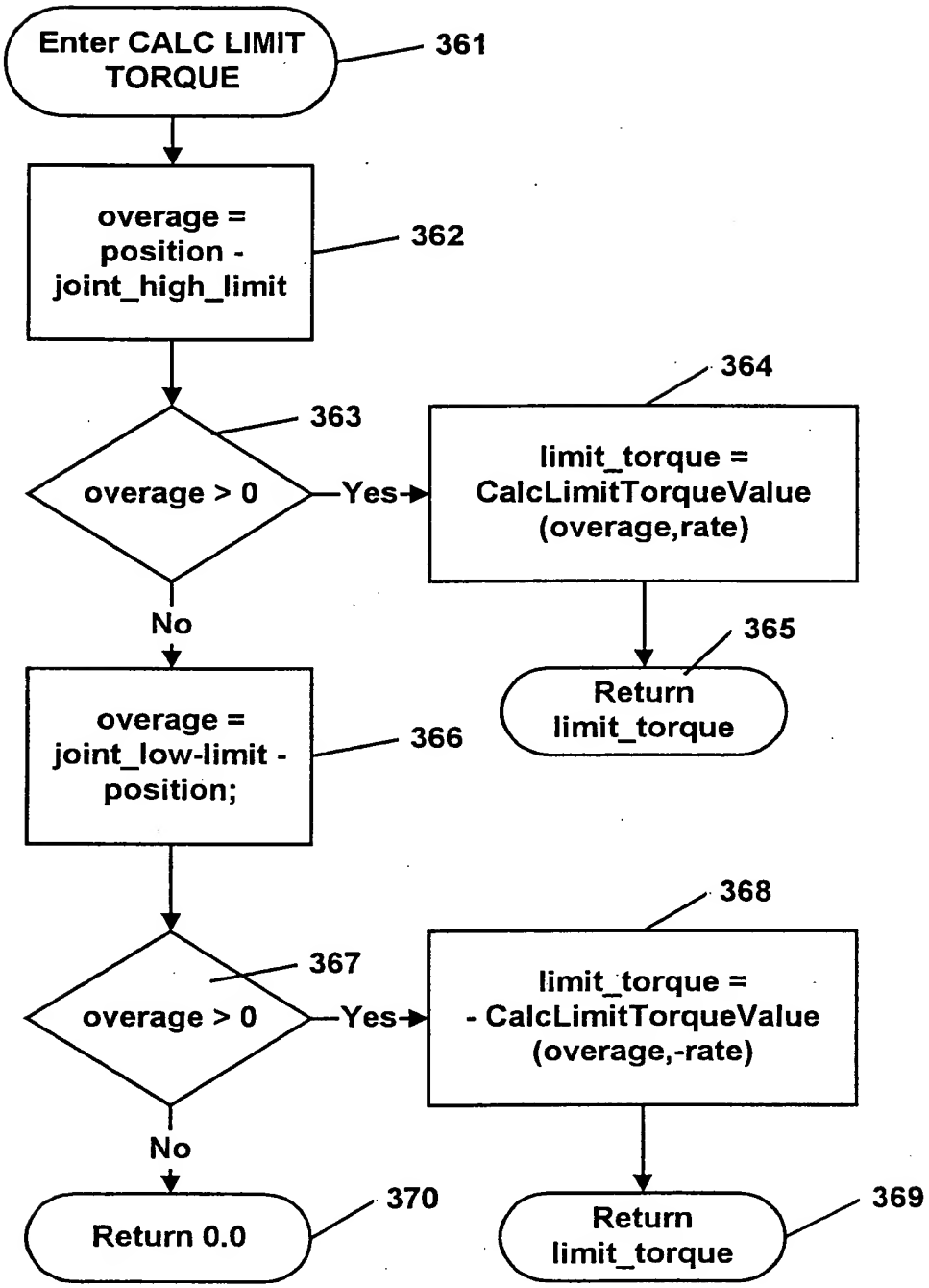
Enter CALC LIMIT TORQUE
361
overage = position - joint_high_limit
362
363
overage > 0
Yes
364
limit_torque = CalcLimitTorqueValue (overage,rate)
No
365
Return limit_torque
overage = joint_low-limit - position;
366
367
overage > 0
Yes
368
limit_torque = - CalcLimitTorqueValue (overage,-rate)
No
370
Return 0.0
369
Return limit_torque

*Fig. 34*

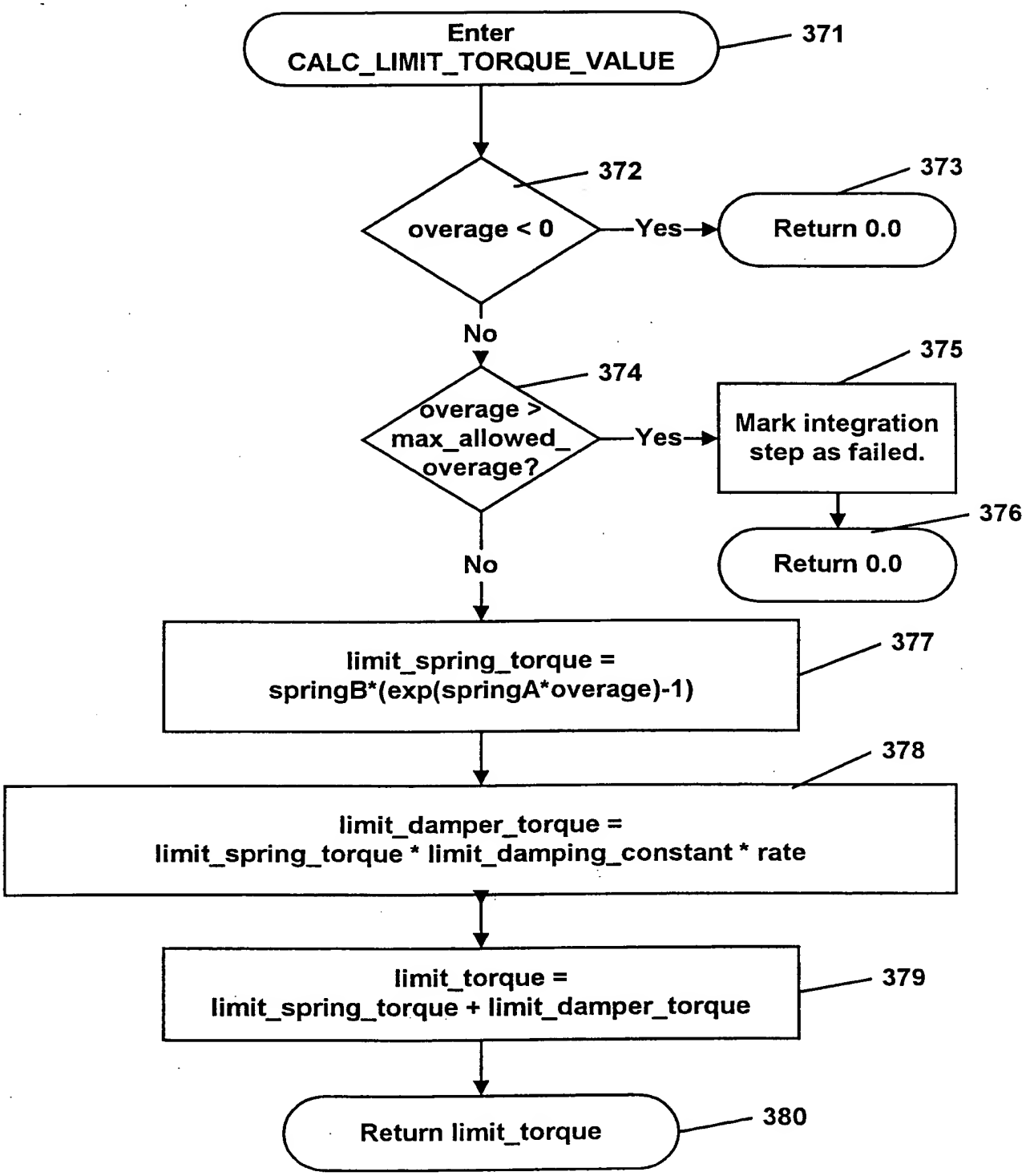
Enter
CALC_LIMIT_TORQUE_VALUE
371
372
overage < 0
Yes
373
Return 0.0
No
374
overage >
max_allowed_
overage?
Yes
375
Mark integration
step as failed.
376
Return 0.0
No
limit_spring_torque =
springB*(exp(springA*overage)-1)
377
378
limit_damper_torque =
limit_spring_torque * limit_damping_constant * rate
limit_torque =
limit_spring_torque + limit_damper_torque
379
Return limit_torque
380

*Fig. 35*

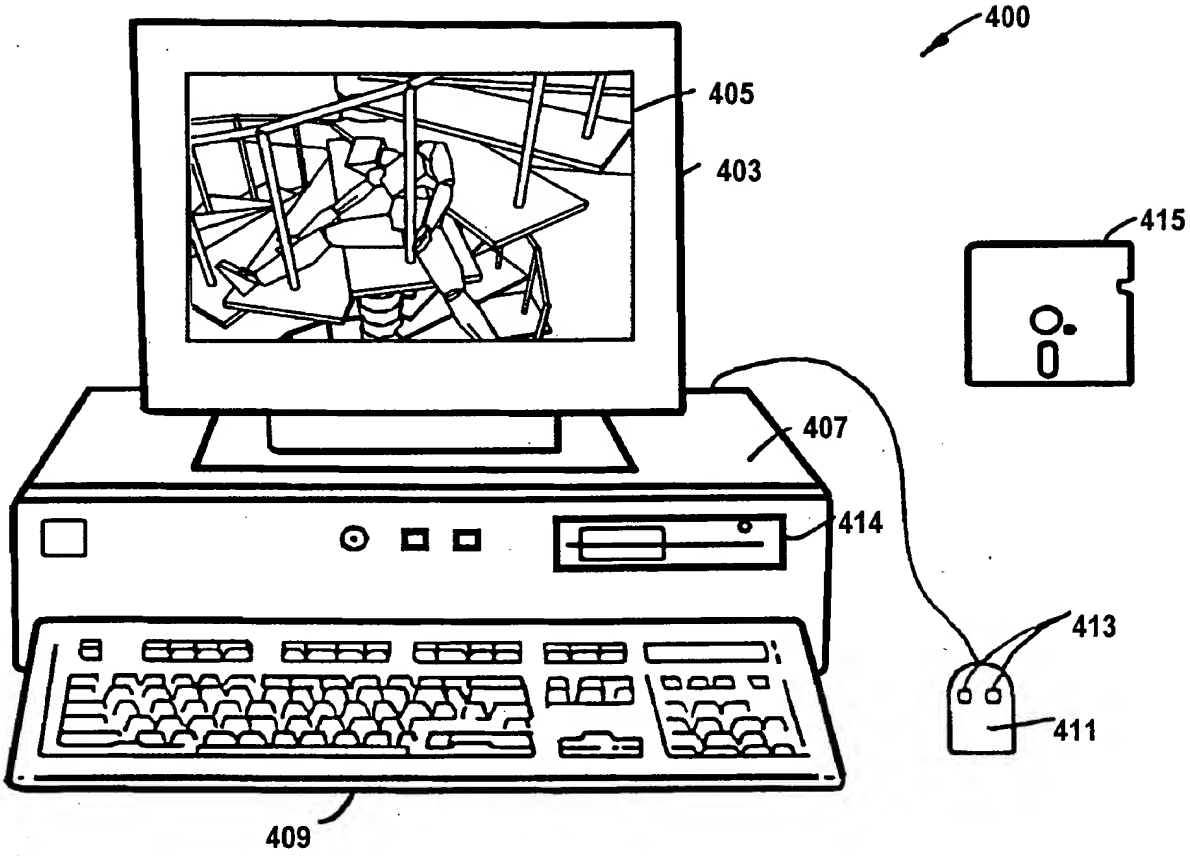


*Fig. 36A*

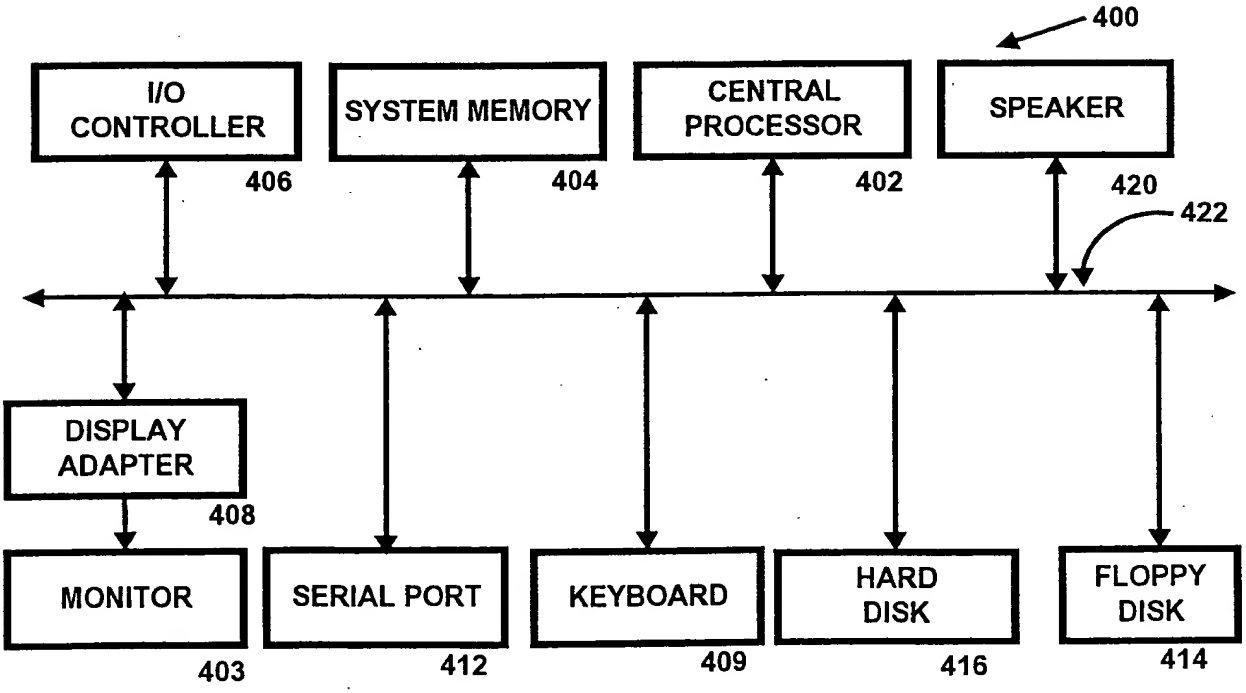


*Fig. 36B*